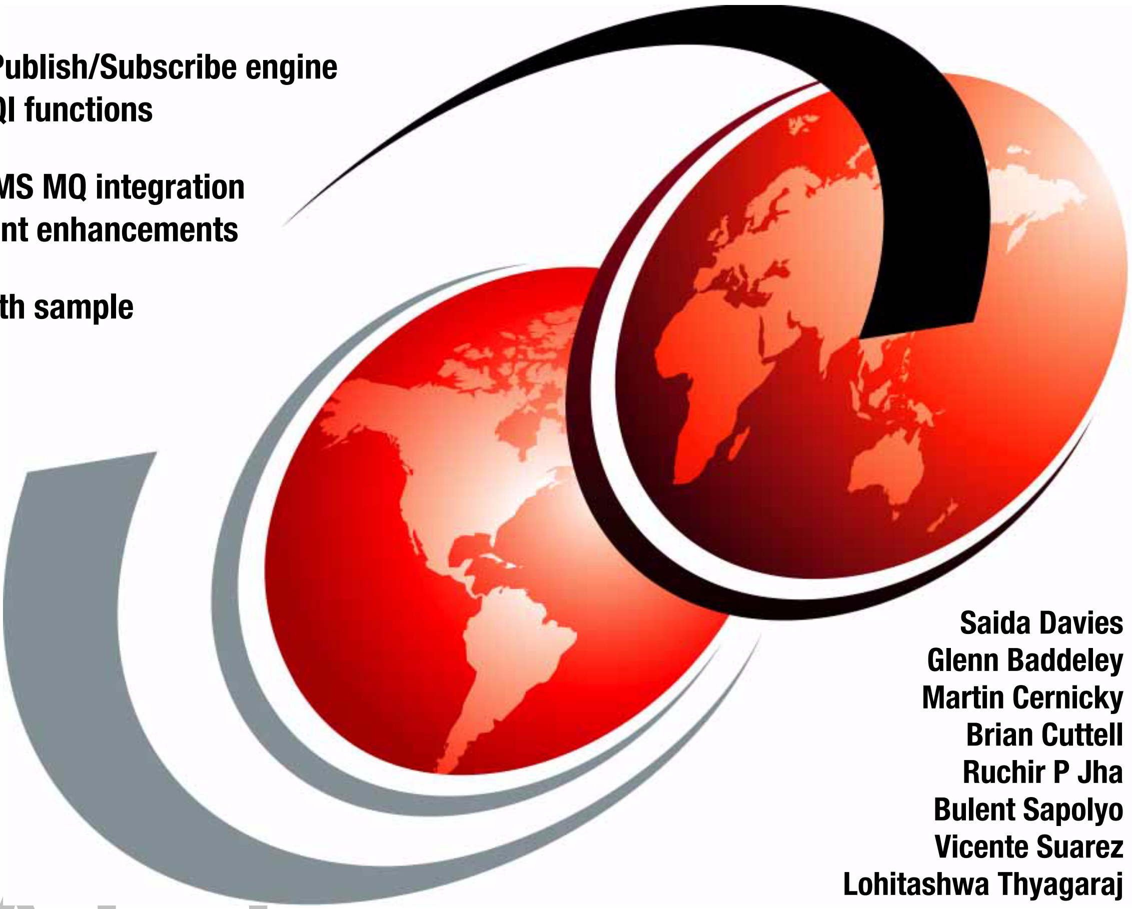
Draft Document for Review April 3, 2008 2:00 pm SG24-7583-00

WebSphere MQ V7.0 Features and Enhancements

Integrated Publish/Subscribe engine and new MQI functions

Improved JMS MQ integration and MQ Client enhancements

Scenario with sample code



Saida Davies
Glenn Baddeley
Martin Cernicky
Brian Cuttell
Ruchir P Jha
Bulent Sapolyo
Vicente Suarez
Lohitashwa Thyagaraj

ibm.com/redbooks

Redbooks

International Technical Support Organization

WebSphere MQ V7.0 Features and Enhancements

March 2007

SG24-7583-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xxi.

Version	Release	Modification	product-name
7	0	0	WebSphere MQ
6	0	2.2	WebSphere MQ Client (SupportPac MQC6)
1	6	0_03	Java JRE (Sun™ Java™ SE Runtime Environment)
6	0	2	Microsoft Internet Explorer
			Microsoft Windows XP Service Pack 2
10	3 (i586)		openSUSE
2	0	0.6	Mozilla Firefox

First Edition (March 2007)

This document created or updated on April 3, 2008.

Note: This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

Contents

Figures

Tables

Examples

Notices

Trademarks

Preface

This book is divided in three parts:

Part 1 commences with an introduction to Message Oriented Middleware and the WebSphere® MQ product. The concept of messaging is covered, explaining what is new in WebSphere MQ V7.0 and how it is implemented. An overview is provided on how it fits within the Service Oriented Architecture (SOA) framework.

Part 2 explains the new WebSphere MQ V7.0 features and enhancements in detail and includes compatibility and the migration considerations from the previous supported versions. The new features and enhancements covered are:

Introducing new WebSphere MQ V7.0 features:

- Both MQI and JMS APIs
- RAS features within JMS

Exploring the new features:

- Publish/Subscribe
  Consolidating pub/sub domain
  Distributed pub/sub
  Available on z/OS®
- MQI enhancements
  Message selectors
  Message properties
  Call-back allows asynchronous consumption
- Client enhancements
  Asynchronous put
  Full duplex
  Conversation sharing
  Read ahead
  Channel instance limits
- Interaction between JMS and MQI Applications
- MQ Explorer enhancements including JMS administration

- MQ HTTP bridge
- z/OS enhancements
- Migration considerations

Part 3 of the book contains the scenario which demonstrates how the new features and enhancements work and how to use them. The sample programs and scripts used for this scenario are available for download by following the instructions in Appendix B, “Additional material” on page 343.

The information included in this IBM® Redbook Publication complements but does not replace product documentation.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Hursley Center.

From left: Saida, Vicente, Lohitashwa, Ruchir, Bulent, Martin, Glenn, Brian

Saida Davies is a Project Leader for the International Technical Support Organization (ITSO) and has extensive experience in Information Technology. She has published several Redbooks® and Redpapers on WebSphere Business Integration, Web Services and WebSphere Service Oriented Middleware using multiple platforms. Saida has experience in the architecture and design of WebSphere MQ solutions, extensive knowledge of z/OS operating system and a detailed working knowledge of both IBM and Independent Software Vendors' operating system software. As a senior IT specialist, her responsibilities included the development of services for WebSphere MQ within the z/OS and Windows® platform. This covered the architecture, scope, design, project management and implementation of the software on stand-alone systems or on systems in a Parallel Sysplex® environment. She has received Bravo Awards for her project contributions. Saida has a degree in Computer Studies and her background includes z/OS systems programming. Saida supports Women in Technology activities and contributes and participates in the their meetings.

Glenn Baddeley is a specialist in WebSphere MQ and has been with IBM Global Technology Services Australia for ten years. He leads the support of WebSphere MQ on several large outsourcing contracts in the Asia/Pacific region. Glenn performs architecture design and consulting, strategic planning, setting standards, writing specialised documentation, tools programming, product installation and complex problem solving for many critical business applications that use WebSphere MQ on a wide variety of platforms. This has given him a deep understanding of the product and a great appreciation of its practical use. Glenn is a member of the IBM MQ technical leadership team for the region and also contributes as a Subject Matter Expert at the global level. Prior to 1997, he worked for a large telecommunications corporation for fourteen years as a systems programmer, designing, developing and supporting customised middleware solutions and ISV Operating System security extensions. Glenn has a Bachelor degree in Computer Science from Deakin University, Australia. He is the author of IBM SupportPac™ MA0K and presented a session on WebSphere MQ Client Security at the IBM Interaction 2000 conference in Australia.

Martin Cernicky is a certified IT Specialist from the IBM Software Services in the Czech Republic. He has seventeen years of experience within the IT sector, and has been working with IBM since 1995. Martin is currently working in the pSeries® support team and has excellent knowledge of AIX/pSeries systems and solutions. Additional responsibilities over the last five years have included supporting WebSphere MQ middleware messaging and WebSphere Message Broker. He has designed and implemented solutions using the WebSphere MQ family products and has substantial experience implementing messaging and broker solutions. Martin is a co-author of IBM Redbooks Publications *Migrating to WebSphere Message Broker Version 6.0*, SG24-7198 and *Managing WebSphere Message Broker Resources in a Production Environment*, SG24-7283. Martin holds a degree in Automated Technology Systems at The Czech Institute of Technology.

Brian Cuttell is a Program Manager working with IBM Betaworks organization in the UK. He has twenty five years of experience in the IT Sector with IBM and other companies. In his career with IBM Hursley Labs, Brian was part of the CICS® development team.and in the early 1990s and was one of the original members of the MQ development team on MVS™. Since then, he has worked for various assignments including IBM UK's graduate program. Brian has more recently specialised in managing Beta programs for a variety of software products. He is currently managing the customer Beta program for WebSphere MQ V7.0. Brian holds a degrees in Mathematics from the University of Oxford, and Operational Research from the University of Lancaster.

Ruchir P Jha Ruchir Jha is a System Software Engineer working in the Application Integration Middleware team for IBM India Software Labs in Bangalore, India. He is responsible for designing scenarios which test the interoperability of WebSphere MQ with other products of the WebSphere Business Integration Suite. These scenarios enable Ruchir to discover and resolve defects, which helps customers avoid similar situations in the future. Strategies created which help customers who are trying to deploy SOA solutions that have WebSphere MQ as a messaging backbone. Ruchir has presented papers at prominent international conferences, and possesses an engineering degree in Computer Science from Nirma Institute of Technology, India.

Bulent Sapolyo is the IBM Asia Pacific Product and Technical Leader for WebSphere MQ, within Global Technology Services (GTS). He has over twenty-five years experience within the IT sector, and has been working with IBM since 1994. Bulent has vast experience in various financial sectors, prior to joining IBM as a Senior Systems Programmer. Since joining IBM, he has held many positions within the firm as an architect, consultant and a technical leader with IBM Asia Pacific for DBDC products, on z/OS and WebSphere MQ, on Midrange, Intel® and Mainframe z/OS platforms. In addition to having experience with various operating systems, Bulent possesses detailed working knowledge of both IBM and Independent Software Vendor (ISV) operating system software. As a Product and Technical Leader with IBM Global Technology Services, his role includes the setting of software direction and migration paths for outsourced customers and implementation of WebSphere MQ. Bulent also designed the architecture, scope and implemented the WebSphere MQ on stand-alone systems, in a Parallel Sysplex environment and with High Availability solutions on Midrange and z/OS platforms.

Vicente Suarez is a Senior IT Specialist working for IBM Hursley in the UK, with eight years of experience as a specialist in WebSphere MQ and WebSphere Message Broker. Vincent has been with IBM since 1988, where he started off working with the IBM Travel and Transportation Industry Team in IBM Colombia. Vincente is co-author of the Redbook publication *WebSphere BI for FN for z/OS V1.1.0 Installation and Operation*, SG24-6090 and various Redpaper publications such as *WebSphere MQ V6, WebSphere Message Broker V6 and SSL*, REDP_4140-00. In addition, Vincente has developed and published support packs for WebSphere Mesasge Broker, and articles in IBM developer works. In his current role as an IBM Software Services for WebSphere consultant, he promotes the use of the practises that best maximize the software performance and facilitate the use, maintenance and operation of WebSphere products for IBM customers' world-wide.

Lohitashwa Thyagaraj is an Advisory Software Engineer working for IBM India Software Labs in Bangalore for the past three years. He has eight years of experience in the IT Industry with expertise in the Banking, Enterprise Application Integration and WebSphere Service Oriented Middleware. Lohitashwa is the Tech lead for the WebSphere Application Server and WebSphere MQ Java/Java™ Message Service Level-3 service team and his key responsibility is to provide technical support for IBM customers worldwide using these products. He also chairs a technical forum called Messaging Technology Council-Bangalore (MTC-B) that is dedicated in nurturing emerging technologies within the Message-Oriented Middleware. Lohitashwa was appointed as one of the Top Talent employees for the year 2007 within the India Software Labs. He holds a degree in Computer Science from Bangalore University.

The ITSO would like to express its special thanks to IBM BetaWorks™, Hursley for hosting this project .

Sincere thanks to:

Brian Cuttell
WebSphere Business Integration Early Programs Test Environment Specialist, IBM Sales & Distribution, Software Sales, IBM Hursley for his support facilitating the residency in Hursley.

Tasnim Kapasi
Gap year student completing her Industry experience with IBM. Her contribution was to set up specific variables, build the glossary and the index incorporated in the book. She reviewed and edited the material required for this book and created the Preface using Adobe® FrameMaker. Tasnim also used Paint Shop Pro to upload and edit photographs into text documents. She has excellent knowledge of all Microsoft® and Apple software. Tasnim has achieved excellent A-levels in English Literature, Economics and Art.

The team would like to thank the following people for their assistance and contributions to this project:

Ben Mann
IBM Software Group, WebSphere MQ product manager, Application and Integration Middleware Software, IBM Hursley, UK

Matthew White
WebSphere MQ JMS Development, Software Group, Application and Integration Middleware Software, IBM Hursley, UK

Morag Hughson
WebSphere MQ Base Architect, Software Group, Application and Integration Middleware Software, IBM Hursley, UK

Andrew Wheal
Betaworks System Support(Hardware), IBM Sales & Distribution, Software Sales, IBM Hursley, UK

Paul Clarke
WebSphere MQ Development, Software Group, Application and Integration Middleware Software, IBM Hursley, UK

Mark A Butcher
WebSphere MQ Development Project lead, Software Group, Application and Integration Middleware, IBM Hursley, UK Software

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

`ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Part 1

Introduction

This part commences with an introduction to Message Oriented Middleware the WebSphere MQ product. The concept of messaging is covered, explaining what is new in WebSphere MQ V7.0 and how it is implemented. An overview is provided on how it fits within the Service-oriented architecture (SOA).

This part consists of:

1

Overview

This book provides an overview on how it fits within the Service-oriented architecture (SOA) framework and introduces the new WebSphere MQ V7.0 features and enhancements. It includes compatibility and the migration considerations from the previous supported versions. The concept of messaging is covered, explaining what is new in WebSphere MQ V7.0 and how it is implemented. The scenario demonstrates how the new features and enhancements work and how to use them.

The following topics are discussed:

- "Executive summary"
- "The scope of this book"
- "Intended audience"
- "What is covered in this book"
- "What is not covered in this book"
- "Assumptions"

1.1 Executive summary

The power of the WebSphere MQ product is its flexibility combined with reliability, scalability, and security. This flexibility provides a large number of design and implementation choices. Making an informed decision from this range can simplify the development of applications and the administration of a WebSphere MQ infrastructure.

Applications that access a WebSphere MQ infrastructure can be developed using a wide range of programming paradigms and languages. These applications can execute within a wide range of software and hardware environments. Customers can use WebSphere MQ to integrate and extend the capabilities of existing and varied infrastructures in the information technology (IT) system of a business.

1.2 The scope of this book

This publication covers the core WebSphere MQ enhancements made in WebSphere MQ V7.0 and concepts that needs to be understood when developing any application that accesses a WebSphere MQ infrastructure.

Broader understanding is key to making informed design and implementation choices for both the infrastructure and the applications that access that infrastructure.

Throughout this publication details of new areas of function for WebSphere MQ V7.0 are introduced such as the WebSphere MQ Explorer, Publish/Subscribe integration, MQ Java Message Service providers and Message Queue Interface extensions, enhancements on administration, MQ Client and z/OS. Migration considerations when moving to WebSphere MQ V7.0 from prior releases are also discussed.

1.3 Intended audience

This book provides detail about the IBM WebSphere MQ V7.0 product features and enhancements required for individuals to make informed application and design decisions prior to implementing a WebSphere MQ infrastructure or begin development of a WebSphere MQ application. This publication is intended to be of use to a wide-range audience.

1.4 What is covered in this book

This book consists of three parts that provides an environment in which to gain an understanding of the WebSphere MQ concepts and the Publish/Subscribe integration with Java Message Service and Message Queue Interface. WebSphere MQ enables new applications to communicate with existing applications without having to change your existing application code, allowing business to function as usual.

Chapters 1 to 3 in Part 1, provide an overview of this publication and introduction to the WebSphere MQ concepts of messaging and how WebSphere MQ V7.0 product can be exploited.

Chapters 4 to 11 in Part 2, discus the new features and enhancements in WebSphere MQ V7.0 which apply to the full function support for Publish/Subscribe features, administration and client enhancements. It also covers relevant changes to other components of WebSphere MQ.

Chapters 12 to 18 in Part 3, illustrate the scenarios giving examples on how to use the features described in Chapters 4 to 10.

1.5 What is not covered in this book

This publication does not focus on interacting with WebSphere MQ using a particular programming language.

This book does not distinguish between WebSphere MQ administrators and WebSphere MQ application programmers.

1.6 Assumptions

Part 1 of this publication gives the reader a basic understanding of messaging middleware technologies and the relationship to IBM products, which requires no previous technical knowledge. However, parts 2 and 3 concentrate on all the new features and enhancements in WebSphere MQ V7.0 and illustrate most of them in a multi-faceted scenario. This assumes good knowledge of many of the basic features which were introduced in previous versions of WebSphere MQ. Refer to the redbook *WebSphere MQ V6.0 fundamentals*, SG24-7128 for a comprehensive introduction to the foundation features of WebSphere MQ.

2

Concepts of messaging

This chapter discusses the main concepts of messaging, the two different messaging paradigms supported by Java Message Service and positioning messaging in Service-oriented architecture.

The following topics are discussed:

- "Enterprise messaging"
- "Introducing Publish/Subscribe"
- "Java Message Service"
- "Position messaging in Service-oriented architecture"

2.1 Enterprise messaging

Enterprise messaging is a mechanism to exchange messages and data between various disparate applications. The main feature of enterprise messaging is that it decouples the applications running, thereby eliminating the need to synchronize applications running on different operating systems provided by various vendors. The common building block of the data getting exchanged is a message and this message can be data, file, events, web services request/response and so on.

Enterprise messaging is in its naive stage and not well known like the Remote Procedure Calls (RPC), Common Object Request Broker Architecture (CORBA) or any other distributed technologies. Unlike in the case of RPC, where it is a synchronous operation, this means that the receiving application runs until the data from the other application is received. Enterprise messaging provides an environment to send and receive message asynchronously.

One of the methodologies to enable asynchronous way of exchanging messages is by using a messaging infrastructure in between the application to send and receive messages, for example, putting a queue to store and forward the messages to the applications.

2.1.1 Message Oriented Middleware (MOM)

A software product which provide enterprise messaging services is usaully called Message Oriented Middleware (MOM).

There are many MOM software products available and WebSphere MQ is the IBM implementation of MOM. An introduction to WebSphere MQ is provided in Chapter 3, “Introduction to WebSphere MQ” on page 21.

As the number of applications participating in the messaging infrastructure increases, a need for standardized way of exchanging of messages is necessary so that two or more applications understand the protocols they need to use to exchange messages. This where Java Message Service (JMS) is utilized. An introduction to JMS is provided in the section 2.3, “Java Message Service” in this chapter.

2.2 Introducing Publish/Subscribe

Publish/Subscribe (Pub/Sub) is a messaging paradigm to send and receive messages asynchronously without the applications having to know where the application is running or whether the publisher/subscriber application is running or not.

2.2.1 Publish/Subscribe

Pub/Sub is intended for situations where a single message is required by, and is distributed to, multiple users. Their big advantage over other delivery methods is that they keep the publisher separated from the subscriber. This means that the publisher in a publish and subscribe application does not need to have any knowledge of either the subscriber's existence or the applications that may use the published information. Likewise, the subscriber or subscriber application does not need to know anything about the publisher application, hence decoupling the dependency on each other. This decoupling of publishers and subscribers allows for greater scalability and a more dynamic network topology.

Pub/Sub is a sibling of the Message Queue paradigm, and is typically one part of a larger Message Oriented Middleware solution. Most messaging systems support in their Application Programming Interface (API), for example, JMS supports both the Publish/Subscribe and Message Queue models. Data replication for sharing of information among various programs running on multiple machines is an example of a distributed execution model that can be applied to Pub/Sub systems to endow them with stronger fault tolerant and consistency guarantees.

The example below explains how Publish/Subscribe model works:

A sports website which is dedicated for providing information on various games (Cricket, Football, Hockey, Rugby….), provides hourly and daily updates on the events happening in the sports field from across the world. There may be several users registered with this website, interested in receiving regular updates on the sports of their choice. Some users may choose to subscribe and receive information for only a specific topic such as Cricket, another user can be interested in both Cricket and Football and another user can be interested in all the topics.

The website providing the hourly or daily updates on the sports event is the Publisher and the users subscribed to this website and consuming the messages are the Subscribers. If there are multiple subscribers registered, it would be very difficult for the publisher to publish individual messages to all the subscribers registered on various topics. Also, since the publisher publishes messages randomly during the day, the subscribers cannot be active all the time. The publisher and the subscribers would like to send and receive the messages asynchronously.

The Pub/Sub model is a classic notion for addressing such combinations. In the Pub/Sub model, the publisher can publish all the messages related to a sports category to a specific topic, such as cricket, without having to know how many subscribers are subscribed or whether the subscribers are active or not. In this case, the publisher has to publish only one message to the topic for every update irrespective of how many subscribers are subscribed.

When the subscriber logs into the website, all the messages published to the topic for which the subscriber has subscribed appear instantly. The same subscriber can subscribe to more then one topic of interest and still get all the messages published for those topics without any dependency on the publisher application.

2.2.2 Message selection

In a typical Pub/Sub model, the publisher publishes message to a topic and there are one or multiple subscribers subscribed to that particular topic to receive the messages. Subscribers generally get the sub-set of the total messages published by the publisher which they are intended for. Subscribers can filter messages in two ways either by topic based or content based selection.

Topic based

In this model the publisher publishes all the messages on to a topic and subscribers subscribing to this topic receive all the messages published on this topic. All the subscribers receive the same copy of the published message.

For example, at the following figure, a publisher publishes a message called *Welcome to world of colors* to the topic Color and all the subscribers that subscribed to that topic receives the same replica of the published message.

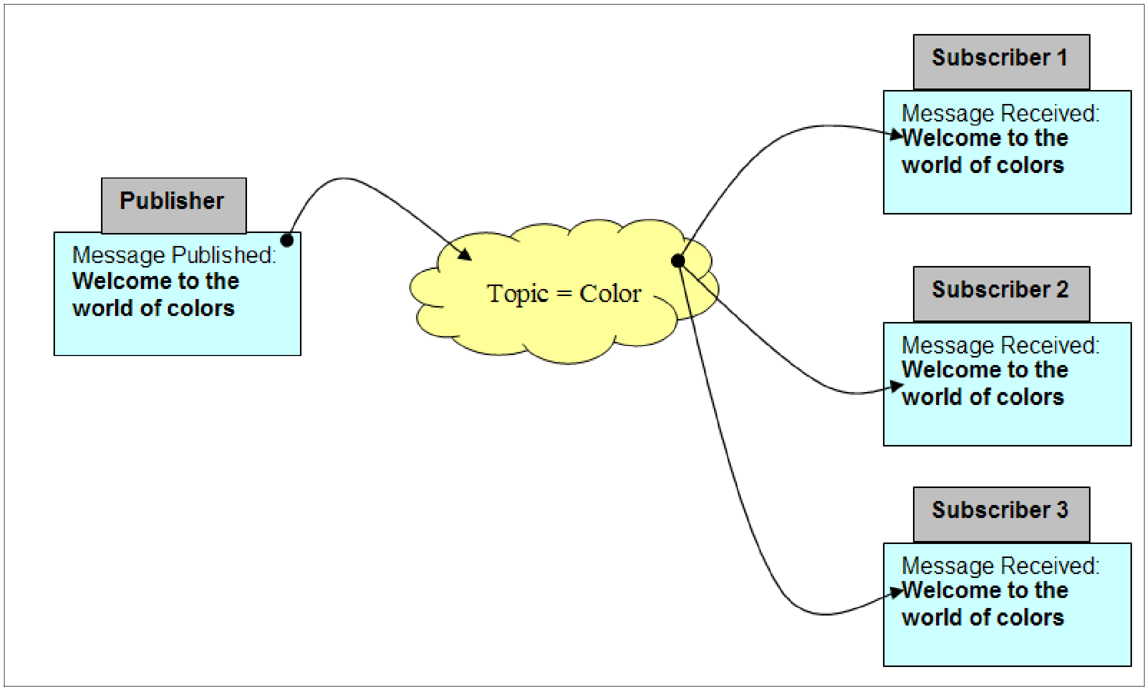

Publisher
Message Published: Welcome to the world of colors
Topic = Color
Subscriber 1
Message Received: Welcome to the world of colors
Subscriber 2
Message Received: Welcome to the world of colors
Subscriber 3
Message Received: Welcome to the world of colors

Figure 2-1 : Topic based selection

Content based

In this model the publisher publishes all the messages on to a topic and the subscribers subscribed to this topic filter the messages they want based on the contents of the message. The subscriber can specify the constraints for the kind of messages they want to receive.

For example, in the following figure, the first publisher publishes a message called *Welcome to the world of colors* and also sets another property on the message as *Color=RED*, the second publisher also publishes the same message but sets the property on the message as *Color=BLUE*. Even though all the subscribers are subscribed to the topic Color, subscribers receive only those messages depending on their constraints. If no constraints are specified then that subscriber gets all messages published to that topic.

Some systems support a hybrid of the two; publishers post messages to a topic while subscribers register content based subscriptions to one or more topics.

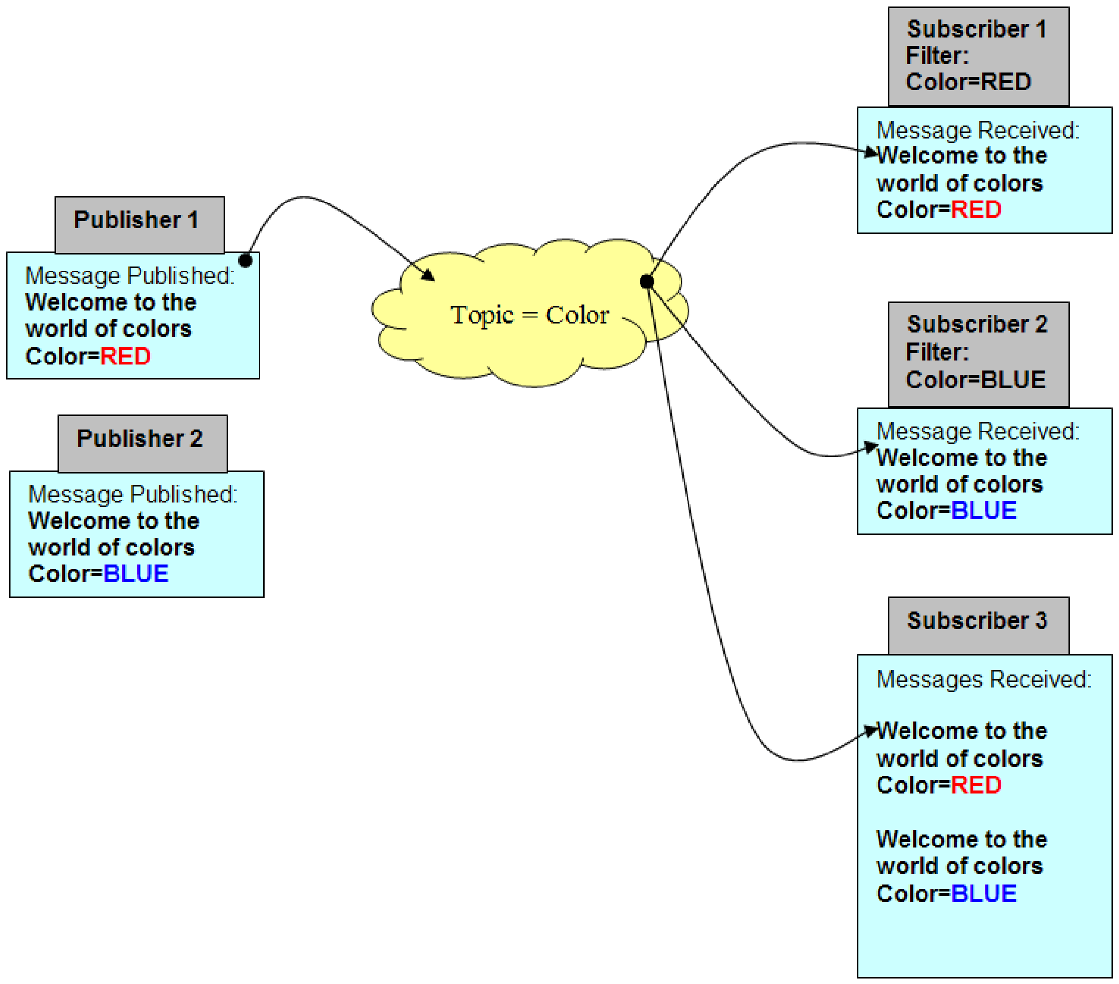


Figure 2-2 : Content based selection

2.2.3 Advantages

Loosely coupled: In the traditional tightly coupled client-server architecture it is mandatory that both the client and server applications must be running if these applications want to exchange messages between each other. The server application cannot receive any messages if the client application is not running and the client application cannot send any messages if the server application is busy processing something else or is down. Therefore, the dependency on both the systems are very high.

In the Pub/Sub model, the applications are loosely coupled, that is, the publisher and the subscriber applications need not know the existence of each other. With the topic being the focus, publishers and subscriber applications are allowed to remain ignorant of system topology and can exchange messages asynchronously across the system.

Scalable: Pub/Sub provides the opportunity for better scalability than traditional client-server, through parallel operation, message caching, tree based or network-based routing and so on.

2.3 Java Message Service

This section discusses Java Message Service concepts and positioning it within the enterprise messaging system.

2.3.1 Java Messaging

Java Message Service (JMS) is a set of interfaces and associated semantics that define how a JMS client application accesses the facilities of an enterprise messaging product. Enterprise messaging is recognized as an essential tool for building enterprise applications and E-commerce systems, and JMS provides a common way for Java programs to create, send, receive, and read an enterprise messaging system's messages.

Rather than allowing the applications to communicate directly with each other, some components of an application that is based around a message service send messages to a message server. The message server, in turn, delivers the messages to the specified recipients. This might seem like an extra, unnecessary layer of software, but the advantages a message service provider has often outweigh the disadvantages. The message service model is much like the model behind the postal service. We can directly deliver our own mail to our friend and relatives, but letting someone else do it for us greatly simplifies our life. The addition of the messaging service adds another layer to the application but it greatly simplifies the design of both the clients and the servers as they are no longer responsible for handling communications issues. It greatly enhances scalability.

JMS defines several interfaces for message services but no particular implementation, this gives a great flexibility for all vendors to implement the way they want but still ensure interoperability happens between various messaging providers when exchanging messages. JMS allows programmers to develop JMS applications that are independent from any JMS messaging providers. This way the programmers do not have to rewrite their whole application when changing the messaging system. JMS do not have any dependency to a particular software vendor. Therefore JMS is vendor neutral and comply with existing products.

JMS offers two different messaging paradigms:

- Point-To-Point
- Publish/Subscribe

2.3.2 Point-To-Point Model

Point-To-Point model is built around the concepts of message queues. The messages are all destined to a queue and the application would connect to that particular queue and retrieve the messages. The destination in the Point-To-Point model is called as a queue which has the capability of storing and forwarding the message to the destined application.

The below figure depicts how a simple Point-To-Point model works:

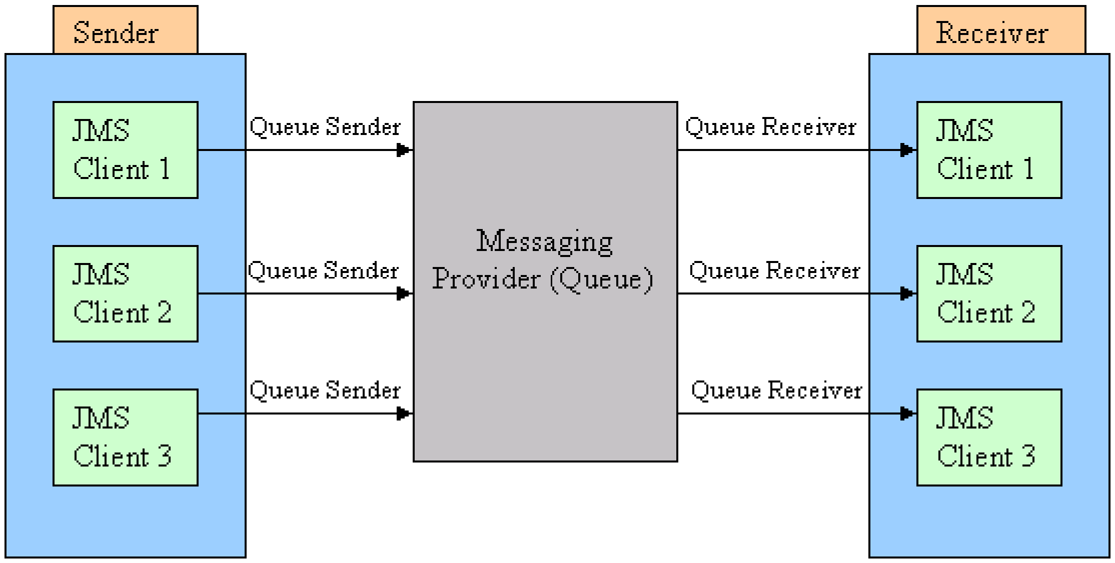


Figure 2-3 : Point-To-Point Model

2.3.3 Publish/Subscribe Model

Pub/Sub model is built around the concept of topics. The destination in the Pub/Sub model is called as a topic. Publisher is an application that sends messages to the topic. Subscriber is an application that receives messages from topic.

See "Introducing Publish/Subscribe" on page 10 for more information.

2.3.4 Advantages of JMS

- JMS allows programmers to write applications independent of the messaging providers, this provides the de-coupling of applications and the messaging provider
- Many IT infrastructures keeps on upgrading, resulting in a change on both the server and client side, because the applications are independent from the messaging providers, the problem of scalability is well addressed by JMS.
- Many enterprise systems resides on multiple platforms, and applications run on various platform, since JMS is purely a Java component the platform in which they reside and the programming language being used is irrelevant as long as the components can understand JMS.

2.4 Position messaging in Service-oriented architecture

Service-oriented architecture (SOA) is a business centric IT architectural approach that supports business integration as linked, repeatable business tasks, or services. SOA helps users build composite applications, which are applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes. SOA is an architectural style that makes this possible.

For more information on SOA refer the following link:

http://www.ibm.com/soa

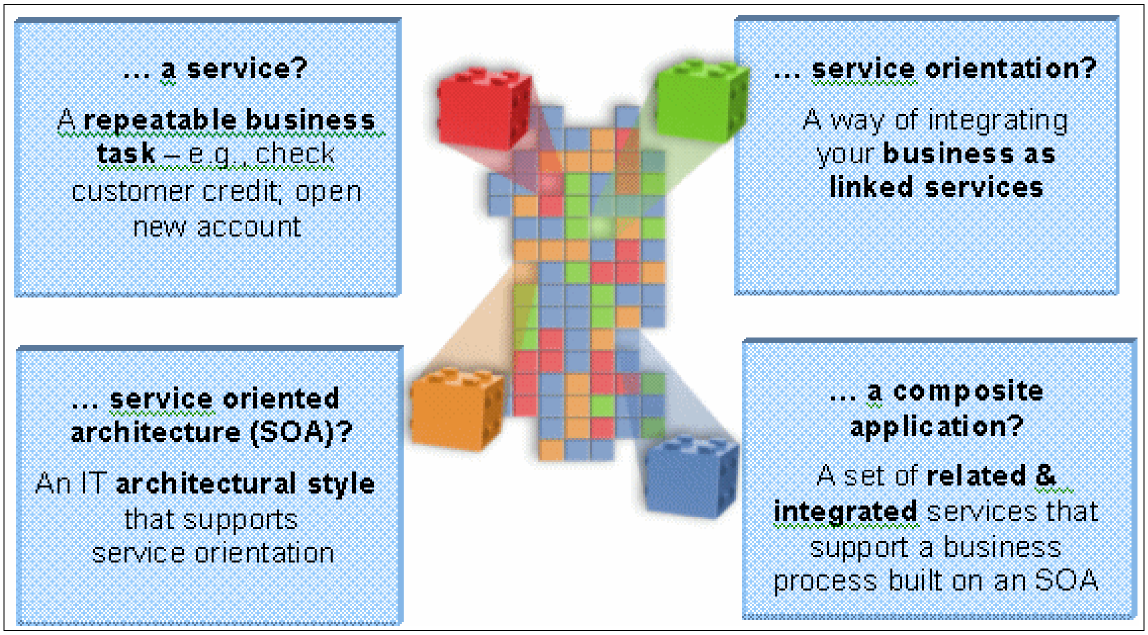

... a service?
A repeatable business task – e.g., check customer credit; open new account
... service orientation?
A way of integrating your business as linked services
... service oriented architecture (SOA)?
An IT architectural style that supports service orientation
... a composite application?
A set of related & integrated services that support a business process built on an SOA

Figure 2-4 : IBM SOA model

The most important characteristic of SOA is the flexibility to treat elements of business processes and the underlying Information and Technology (IT) infrastructure as secure, standardized components (services) that can be reused and combined to address changing business priorities. The key feature of SOA is the ability for various applications to interact with each other.

In April 2006, IBM defined the following key SOA entry points based on real customer experiences, customer engagements and allowed customer the flexibility to choose any of entry point based on their necessity and still achieve SOA:

- People
- Process
- Information
- Reuse
- Connectivity

In today world, connectivity is no longer a mere link between two or more machines. Service connectivity is an IT centric entry point to SOA that is designed to help simplify IT environment. It provides a more secure, reliable and scalable way to connect within and beyond the business. SOA links people, processes and information with a seamless flow of messages and information from virtually anywhere, at anytime and using anything. It brings new levels of flexibility to such linkages and delivers real business value on its own. Connectivity is also a core building block for future SOA initiatives.

As more and more applications participate in the SOA environment, the need to exchange data and messages also arises, thereby requiring a more robust and reliable way for exchanging messages. Hence, making messaging backbone as the foundation for SOA connectivity.

The above figure shows the International Business Machines Corporation (IBM) representation of Enterprise Service Bus in relation to the reference architecture, and shows where WebSphere MQ fits into that architecture. WebSphere MQ is a core component in the enterprise service bus providing reliable communication between applications. While service orientation gives a new focus to thinking about reuse of software components, this is not a new thing because WebSphere MQ has been providing fast, reliable connections between applications for over several years. The concept of Enterprise Service Bus allow integration of new and existing services using proven technology.

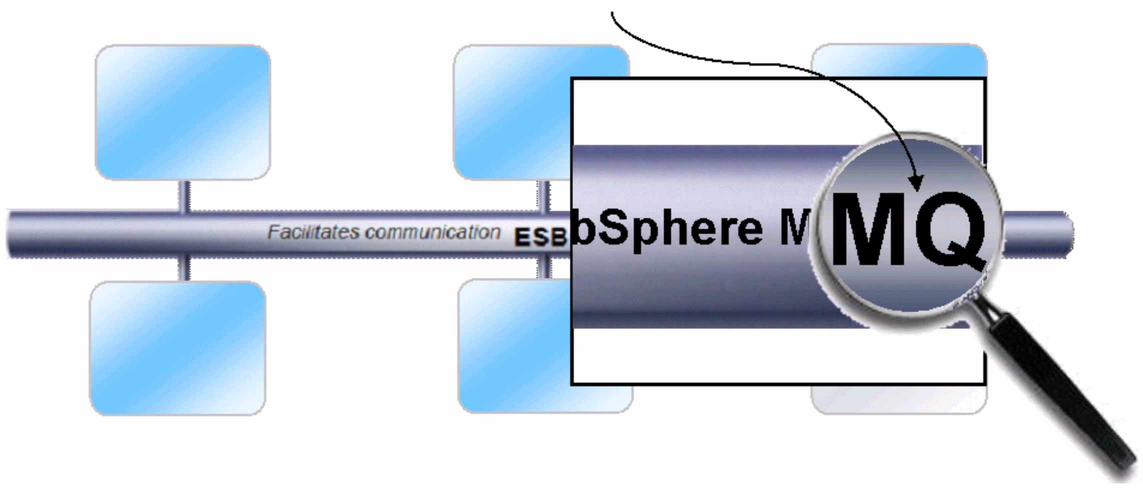


Figure 2-5 WebSphere MQ with relation to reference architecture

As connectivity was identified to be one of the key entry point for SOA, more sub points were added to strengthen the connectivity infrastructure, below are the 7 main points identified for connectivity:

- Reliable
- Secure
- Time flexible and resilient

- Transactional
- Incremental
- Ubiquitous
- Basis for Enterprise Service Bus

In summary, any messaging component fitting in the connectivity infrastructure has to satisfy and support all the above mentioned sub points. Clearly messaging has become a backbone and one of the key component for Service-oriented architecture and Enterprise Service Bus and is responsible for delivering the messages between various applications in a more robust, reliable and secured way.

3

Introduction to WebSphere MQ

This chapter introduces WebSphere MQ, how messaging is implemented with WebSphere MQ, what is new in WebSphere MQ V7.0, and how it fits in the WebSphere product family.

Three sections are presented in this chapter:

- "Messaging with WebSphere MQ"
- "What is new in WebSphere MQ V7.0"
- "Positioning in WebSphere product family"

3.1 Messaging with WebSphere MQ

WebSphere MQ is an established and reliable messaging integration middleware product. Over more than 15 years WebSphere MQ (or MQSeries® as it was known in earlier versions) has grown to provide flexible and reliable solutions which address the wide range of requirements introduced in the previous chapter.

A message queuing infrastructure built on WebSphere MQ technology provides an available, reliable, scalable, secure and maintainable transport for messages with assured once-only delivery.

Many enhancements have been added to WebSphere MQ during its evolution in the market place, including:

- WebSphere MQ Clients: Enables an application to connect remotely or locally to a WebSphere MQ queue manager.
- Publish/Subscribe: Increases messaging capability from point to point messaging to a less coupled style of messaging.
- MQ Clusters: Allow multiple instances of the same service to be hosted through multiple queue managers, to enable load-balancing and simplify administration.
- Secure Socket Layer support: SSL protocol can be used to secure communication between queue managers or MQ Client.
- Diverse platforms: WebSphere MQ supports wide range of operating system platforms.

Note: To learn and discover the essential features and capabilities of WebSphere MQ, refer to IBM Redbook publication *WebSphere MQ V6 Fundamentals*, SG24-7128 available at:

http://www.redbooks.ibm.com/abstracts/sg247128.html?Open

3.1.1 Core concept of WebSphere MQ

Data is transferred between applications in messages. A message is a container consisting of two parts:

- Message Descriptor: Identifies the message and contains additional control information such as the type of message and the priority assigned to the message by the sending application.

- Message Data: Contains the application data. The structure of the data is defined by the application programs that use it and MQ is largely unconcerned with its format or content.

The nodes within a WebSphere MQ message queuing infrastructure are called queue managers. The queue manager is responsible for maintaining messages. Multiple queue managers can run on a single physical server or on a wide network of servers across a large variety of different hardware and operating system platforms.

Each queue manager provides facilities for reliable message queuing using both point to point and Publish/Subscribe styles.

The queue manager maintains queues of all messages which are waiting to be processed or routed. Queue managers are tolerant of failures and maintain the integrity of the business-critical data flowing through the message queuing infrastructure.

The queue managers within the infrastructure are connected via logical channels. Messages automatically flow across these channels, from the initial producer of a message to the eventual consumer of that message, based on the configuration of the queue managers in the infrastructure. Changes can be made to the configuration of queues and channels, and this is transparent to the applications.

Asynchronous messaging

Two applications which need to communicate, whether hosted on the same machine or separate machines, may have originally been designed to do so directly and synchronously.

In this case the two applications exchange information by waiting for the partner application to become available, and then send information. If the partner application is unavailable for any reason, including if it is busy performing communication with other applications, then the information cannot be sent.

All intercommunication failures which can occur between the two applications, which may be on the same machine or different machines connected by a network, must be considered individually by the applications. This requires a protocol for sending the information, confirming receipt of the information, and sending any subsequent reply.

Placing a WebSphere MQ infrastructure between the two applications allows this communication to become asynchronous. One application places information for the partner in a message on a WebSphere MQ queue, and the partner application processes this information when it is available to do so. If required, It can then send a reply message back to the originator. The applications do not need to be concerned with communication failures or recovery.

WebSphere MQ Clients

WebSphere MQ client is a light-weight component of WebSphere MQ which does not require the queue manager run-time code to reside on the client system. It enables an application running on the same machine as the client to connect to a queue manager that is running on another machine, and issue MQI calls to that queue manager. Such an application is called a client and the queue manager is referred to as a server.

Using MQ Clients is an effective way of implementing WebSphere MQ messaging and queuing. The benefits of doing this are:

- There is no need for a licensed WebSphere MQ server installation on the client machine.
- Hardware requirements on the client system are reduced.
- System administration requirements on the client system are reduced.
- An application using at MQ Client can connect to multiple queue managers on different machines.

Important: Because there is synchronous communication between the client and queue manager, MQ Client requires a reliable and stable network!

Application Programming Interfaces (APIs)

Applications can use WebSphere MQ via several programming interfaces.

Message Queue Interface

The basic interface is the Message Queue Interface (MQI). The MQI consists of the following:

- Calls through which programs can access the queue manager and its facilities.
- Structures that programs use to pass data to, and get data from, the queue manager.
- Elementary data types for passing data to, and getting data from, the queue manager.

Many programming languages are supported depending on the software and hardware platform, for example C, Java and most other popular languages.

Standardized APIs

Utilizing a standardized API can add additional flexibility when accessing services through a message queuing infrastructure. This book uses the term standardized API to represent APIs which are not proprietary to an individual product, such as WebSphere MQ.

Examples of standardized APIs, which can be used to access services provided through a WebSphere MQ infrastructure, are as follows:

- Java Message Service (JMS).
- IBM Message Service Client (XMS).

Wide adoption of these APIs can occur across multiple products. For example, the JMS API is an industry standardized API for messaging, within the Java 2 platform Enterprise Edition (J2EE™) specification.

Note: For information about JMS refer to the section 2.3, “Java Message Service” on page 15.

Reliability and data integrity

The intercommunication performed between queue managers, across channels, is tolerant to network communication failures and WebSphere MQ assures exactly once-only delivery of messages.

Persistent and non-persistent messages

Messages containing critical business data, such as receipt of payment for an order, should be reliably maintained and must not be lost in the event of a failure.

On the other hand, some message may only contain query data, where the loss of the data is not crucial because the query can be repeated. In this case, performance may be considered more important than data integrity.

To maintain these opposite requirements WebSphere MQ uses two type of messages, persistent and non-persistent:

- Persistent messages: WebSphere MQ does not lose a persistent message through network failures, delivery failures, or restart of the queue manager.

  Each queue manager keeps a failure-tolerant recovery log of all actions performed upon persistent messages. This is sometimes referred to as a journal.
- Non-persistent messages: WebSphere MQ optimizes the actions performed upon non-persistent messages for performance.

Non-persistent messages can be lost if network communication between queue managers fails, a queue manager is restarted, or a software failure occurs which ends a queue manager abnormally.

Units of work

Many actions performed by an application cannot be considered in isolation. An application may need to send and receive multiple messages as part of one overall action. Only if all of these messages are successfully sent or received, should any messages be sent or received.

An application which processes messages may need to perform coordinated work against other resources as well as the WebSphere MQ infrastructure. For example, it may perform updates to information in a database, based upon the contents of each message. The actions of retrieving the message, sending of any subsequent reply, and updating the information in the database, must only complete if all actions are successful.

These actions are considered to be within a unit of work (UOW). Units of work performed by applications accessing a WebSphere MQ infrastructure can include sending and receiving messages as well as updates to databases. WebSphere MQ can coordinate all resources to ensure that a unit of work is only completed, if all actions within that unit of work complete successfully.

Note: WebSphere MQ can also participate in units of work which are coordinated by other products. For example, actions against a WebSphere MQ infrastructure can be included in units of work which are coordinated by WebSphere Application Server and DB2®.

3.1.2 WebSphere MQ messaging styles

There are two basic types of messaging style, point to point and Publish/Subscribe.

Point to point

The point to point style is built around the concept of message queues. Messages are stored on a queue by a source application and the destination application retrieves the messages. This provides the capability of storing and forwarding messages to the destination application in an asynchronous or decoupled manner. Synchronous messaging interfaces can also be implemented using point to point.

The source application needs to know the destination of the message. The queue name could be usually enough but in some complex hub and spoke environments the name of remote queue manager is also needed.

Publish/Subscribe

WebSphere MQ Publish/Subscribe (Pub/Sub) allows the provider of information to be decoupled from the consumers of that information. It became available as a SupportPac for WebSphere MQ V5.3 and V6.0.

Before a point to point application can send some information to another application, it needs to know something about that application. For example, it needs to know the name of the queue to which to send the information, and it might also need a queue manager name. Pub/Sub removes the need for the source application to know anything about the destination application. All it has to do is send information it wants to share to a standard destination which managed and distributed by WebSphere MQ. Similarly, a destination application does not need to know anything about the source of the information it receives.

Note: For information about Pub/Sub refer to the section 2.2, “Introducing Publish/Subscribe” on page 10.

3.1.3 WebSphere MQ distributed messaging

A single queue manager topology with its applications running on the same machine has limitations of scale. MQ Client can allow applications to run on remote machines but connections between the applications and the queue manager requires a reliable and stable network which is not usually available for long distances.

This single queue manager limitation can be eliminated without alteration to the applications by using distributed messaging. Applications accessing a service can use a queue manager hosted on the same machine, providing a fast connection to the infrastructure. MQ channels provide transparent asynchronous messaging with service applications hosted by other queue managers on remote machines.

There are two types of distributed messaging: Hub and Spoke, and MQ Clustering.

Hub and Spoke

Applications accessing a service connect to their local queue manager, either as clients (over a fast reliable network) or run on the same machine as local queue manager. For instance, an application in a branch office needs to access a service at headquarters. Asynchronous communication occurs through a spoke queue manager to the service application hosted on a hub queue manager.

The machine which hosts a hub queue manager can host all the applications providing the central services, for instance headquarters applications and databases.

This type of architecture is developed by manually defining the routes from the spoke queue managers to the hub queue manager or to many hub queue managers. Multiple services provided by the infrastructure may be hosted on different hub queue managers, or through multiple queues on the same hub queue manager.

MQ Cluster

A more flexible approach is to join many queue managers together in a dynamic logical network called a queue manager cluster. Queue manager clusters allow multiple instances of the same service to be hosted through multiple queue managers.

Applications requesting a particular service can connect to any queue manager within the queue manager cluster. When applications make requests for the service, the queue manager to which they are connected automatically workload balances these requests across all available queue managers which host an instance of that service.

This allows a pool of machines to exist within the queue manager cluster, each hosting a queue manager and the applications required to provide the service. This is especially useful in a distributed environment, where capacity is scaled to accommodate the current load through multiple servers, rather than one high capacity server. A server can fail or be shut down for maintenance and service is not lost.

Using MQ Cluster technology can also simplify administration tasks because message channels for application data transport are maintained by the cluster and do not have to be explicitly created.

3.1.4 SSL support

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are industry standardized technologies which provide assurance of identity and data privacy for MQ client applications which connect to queue managers via a communications network infrastructure, and also queue manager to queue manager distributed queueing and clustering via a network.

SSL and TLS provide similar capabilities and build upon similar principles for establishing identity. TLS is often considered the successor of SSL, as it provides some enhanced security features. SSL or TLS can be utilized for all communication performed over a network within a WebSphere MQ infrastructure.

Using these technologies, WebSphere MQ can verify the identity of applications connecting to a queue manager, and can also verify the identity of other queue managers within the infrastructure with which it exchanges messages.

Any communication over a network within a WebSphere MQ infrastructure, for which identity has been verified using SSL or TLS, can then be encrypted using a variety of different algorithms within the SSL and TLS standards. This assures the privacy of that communication.

3.1.5 Diverse platforms

WebSphere MQ provides simplified communication between applications running on different hardware platforms and operating systems and implemented using different programming languages.

This enables a business to choose the most appropriate infrastructure components for implementing or accessing services within their system. The messaging infrastructure understands differences between the underlying hardware and software on which individual nodes are running.

Some conversion of character and numeric data might be required in order for the data to be readable across different hardware and software platforms. The messaging infrastructure can be configured to perform this conversion transparently so that each message is valid when it is retrieved at the destination.

A consistent WebSphere MQ implementation exists across a range of more than 80 supported operating environments, provided both by IBM and business partners. To learn about supported platforms for WebSphere MQ refer to the following IBM web pages:

- *WebSphere MQ Platforms & Clients*, available at:

  `http://www.ibm.com/support/docview.wss?rs=171&uid=swg27007431`

- *WebSphere MQ System requirements*, available at:

  `http://www.ibm.com/software/integration/wmq/requirements`

3.2 What is new in WebSphere MQ V7.0

This section briefly describes the new features and enhancements in WebSphere MQ V7.0. Detailed information about the following topics is covered in Part 2, “WebSphere MQ V7.0 enhancements and changes” on page 43 of the book.

3.2.1 Publish/Subscribe integration

WebSphere MQ V7.0 now has tightly integrated Publish/Subscribe messaging, to simplify its configuration, development and deployment. This makes it easier than ever to use Publish/Subscribe to increase the flexibility of messaging solutions. Pub/Sub management is now fully incorporated into the graphical WebSphere MQ Explorer tooling making it easier to use and configure.

WebSphere MQ Publish/Subscribe in V7.0

In WebSphere MQ V6.0 a broker component that was external to the queue manager managed the Publish/Subscribe functionality and needed to be started separately. One broker would be created for each queue manager and used the same name as the queue manager. This broker utilized WebSphere MQ facilities to manage interactions between publishing and subscribing applications.

WebSphere MQ V7.0 provides a new Pub/Sub engine which is integrated into the queue manager and is automatically enabled. The queue manager receives messages from publishers and subscription requests from subscribers for a range of topics. It is then solely responsible for routing the published messages to the target subscribers.

Topics in WebSphere MQ V7.0

A topic refers to the subject on which publishers provide information. Subscribers interested in information on this topic can either subscribe to a topic object or a topic string to receive these publications. With WebSphere MQ V7.0 you can now publish directly to topics.

The topic string is the central concept in WebSphere MQ V7.0 Pub/Sub because it connects publishers and subscribers together. Publishers can now publish to a topic string and subscribers can subscribe to these publications using this topic string. The topic string can be up to 10,000 characters long. A new data type called the Variable Length String has been introduced in WebSphere MQ in order to support this requirement. The structure and semantics of the topic string is controlled by the slash (/) to build the topic hierarchy.

Topic security

Topics inherit security attributes from the nearest administration node using a delegation model. Administration nodes have an associated WebSphere MQ Topic object which defines the security levels of access to topics.

Message selectors

Message selectors can be also used in Pub/Sub messaging. It enables a Pub/Sub application to specify the messages that it is interested in by specifying criteria on message properties. Details about message selectors are described in the section 3.2.3, “MQI extensions”.

Distributed Publish/Subscribe

Distributed Pub/Sub enables applications connected to separate queue managers to use Pub/Sub messaging. There are two types of topologies for distributed Pub/Sub: Hierarchical and Pub/Sub Clusters (also known as Collectives).

A Pub/Sub cluster uses MQ Cluster technology for connectivity of queue managers. In a MQ Cluster, all queue managers are interconnected with each other. A MQ Cluster becomes a Pub/Sub cluster (Collective) by the definition of a clustered topic within the cluster. Although the clustered topic is created on one queue manager, the definition is shared with all queue managers in the cluster, as would occur for ordinary clustered queues. All publications that are made to the clustered topic are sent to all the queue managers in the cluster and are thus sent to the all subscribing applications connecting to each queue manager in the cluster.

A Hierarchical Pub/Sub topology is built on queue managers connected together via standard distributed message channels. A parent and child relationship is defined to build the hierarchy. The hierarchical topology uses either a proxy-subscription or publish-everywhere routing mechanism to deliver messages to subscribers. Thus it can take some time for the subscription to propagate around all queue managers in the network.

3.2.2 WebSphere MQ Client enhancements

WebSphere MQ V7.0 introduces a new quality of service to improve WebSphere MQ client applications and to provide better control of server-connection channel resources.

Full duplex client channels

The TCP/IP transport protocol for MQ Client and JMS MQ Provider is now full duplex. Network failures are detected earlier by allowing independent heartbeats to be performed in each direction on channels. Channel stop requests from the queue manager can now be processed immediately.

Conversation sharing

MQ Client and JMS MQ Provider now provide the capability for threads to share a MQ channel instance. This effectively reduces the number of running channels on the queue manager, making more efficient use of resources.

Read ahead

MQ Client and JMS MQ Provider can now read ahead non-persistent messages into local memory prior to them being requested by the program. This reduces the number of interactions with the queue manager and improves throughput in some circumstances.

Asynchronous put

MQ programs can now put messages to queues without waiting for a response from the queue manager. The status response can be obtained after a set of messages has been put. When used with the MQ Client or JMS MQ Provider, this reduces the number of interactions with the queue manager and improves throughput in some circumstances.

Instance limits on SVRCONN channels

The SVRCONN type channel has been enhanced to add parameters which limit the number of concurrently running instances of the client channel for all connections and connections from each system. This can prevent client programs from running the maximum number of channel instances available on a queue manager and possibly denying other types of channels from starting.

Weighted selection on CLNTCONN channels

New parameters have been added to the CLNTCONN type channel to allow connection to wild-carded queue managers based on a random selection with relative weightings. This provides a simple work-load balancing feature across multiple queue managers.

3.2.3 MQI extensions

WebSphere MQ V7.0 has improved and extended the Message Queue Interface (MQI), particularly to support Publish/Subscribe and to offer similar features to those found in the Java Message Service (JMS) Application Programming Interface.

Variable Length Strings

The MQI now supports variable length strings and some of the new data structures use them. There is a new data type called MQCHARV that is used to represent variable length strings such as topic strings, object names, subscriber user data, selection strings and other new data elements.

Message properties and message handles

Message properties are now supported in the MQI, providing access to name and value pairs which are associated with MQ messages. It is possible to set, inquire and delete the message properties of MQ messages. Message properties can be used to filter messages that are retrieved from a queue or a subscription using message selectors. Message properties require a message handle to refer to them as part of a message. Application can use new MQI function calls to create and delete message handles (MQCRTMH and MQDLTMH) and to set, inquire or delete message properties (MQSETMP, MQINQMP and MQDLTMP).

Message browsing

Message browsing has been enhanced in WebSphere MQ V7.0. New options in MQOPEN and MQGET introduces increased flexibility when browsing messages on queues.

Message tokens are now available to distributed queue managers. Message tokens uniquely identify a message on a queue. A token is returned in the Get Message Option (MQGMO) data structure after a MQGET-with-browse call.

Browse and mark is one of the new MQGET options that enables the queue manager to keep track of which messages have been browsed and by who.

Cooperative dispatchers is a new concept for groups of applications which browse the same queue and that are not interested in messages that have been previously browsed by another cooperative dispatcher.

Call back for asynchronous consumers

Call back is a new set of MQI function calls which enable consumption of messages from queues or subscriptions without using a MQGET-with-wait call. Call back allows implementation of a programming style for message driven processing. Programs using call back functions are called asynchronous consumers.

Asynchronous consumer applications need to register functions that are called back by the queue manager when messages are available and they match the selection criteria. The new MQI calls are MQCB and MQCTL, to register and control call back functions.

Publish/Subscribe

Publisher applications can use MQOPEN and MQPUT, or MQPUT1, to publish messages to topics.

Subscriber applications can create subscriptions to topics using the new MQSUB call or they can request services from a subscription using the new MQSUBRQ call, such as retrieving retained publications on demand. MQGET or call back can be used to retrieve messages from a subscription.

MQSUB has options to create durable or non-durable subscriptions and to specify if the subscriptions are managed or non-managed.

Put action indicators

In WebSphere MQ V7.0 it is now possible to indicate to the queue manager the type of MQPUT or MQPUT1 action that is performed, and the relationship of a new message and a possible original message that has been previously received.

The types of put actions are: new, forward, reply and report. These indicators enable the queue manager to validate and set message properties according to the action.

Message selectors

Message selectors allow an application to specify that it is only interested in receiving particular messages from queues or subscriptions. Only messages whose headers match the filter criteria in the selector are delivered to the application.

Message selectors act on the message properties and headers in a message. The message selector string syntax is based on a subset of SQL92 conditional expressions.

Message selectors exist in JMS and they now are supported by MQOPEN and MQSUB calls.

3.2.4 WebSphere MQ JMS provider implementation

WebSphere MQ V7.0 has extended the Java Message Service (JMS) provider implementation to support new features and to be able to offer similar functionality as the MQI.

Read ahead

The read ahead feature in WebSphere MQ V7.0 allows messages from destinations to be sent to the JMS client ahead of the application actually requesting the messages. This saves the client from having to send a separate request to the WebSphere MQ server for each message it consumes, and allows the client to receive messages in a continuous stream.

Asynchronous put

A JMS client application, for example, responsible for capturing information on climatic changes in humidity, temperature and air pollution, sends sequences of messages in rapid succession to the destination. The client application does not require any immediate acknowledgement of success or reply back for every message sent. After the sequence of messages has been sent the client can confirm that they were all accepted by WebSphere MQ.

Asynchronous consume

WebSphere MQ V7.0 supports both synchronous and asynchronous message consumption. When a JMS application needs to consume message asynchronously it can register a call back function for a destination. When a suitable message is sent to the destination, the function is called and it is passed the message as a parameter. The function can then process the message asynchronously.

Asynchronous consumption of messages by JMS applications was already present in previous releases of WebSphere MQ when JMS applications implemented a JMS MessageListener. However, with WebSphere MQ V7.0, the MQ implementation for JMS has been enhanced to take advantage of the call back mechanism available in WebSphere MQ.

Conversation sharing

Conversation sharing is a new feature in WebSphere MQ V7.0. It allows a single TCP/IP socket to multiplex or share multiple connections or sessions, provided the two ends of the connection belong to the same process. By default, all JMS applications use conversation sharing without any client code modifications.

Mapping of WebSphere MQ and JMS messages

A JMS client application can use message selectors to filter for suitable messages from the destination. The application receives only those messages containing properties matching the specified selector string. The selection is performed by the queue manager.

In WebSphere MQ V6.0 the queue manager did not support message selection natively. The JMS MQ client had to browse the queue sequentially and perform the selection of messages itself. This was very inefficient across a communications network and induced high CPU usage on both the client and the server side when there a significant number of messages on the destination.

Properties of WebSphere MQ classes for JMS

All objects in WebSphere MQ classes for JMS have properties. Different properties apply to different object types. Different properties have different allowable values, and symbolic property values differ between the administration tool and program code.

WebSphere MQ classes for JMS provides facilities to set and query the properties of objects using the WebSphere MQ JMS administration tool, WebSphere MQ Explorer, or in an application. Many of the properties are relevant only to a specific subset of the object types.

3.2.5 Administration enhancements

The MQ Explorer was enhanced in many ways to simplify and provide more secure WebSphere MQ administration.

Other significant changes were made to commands to support new MQ object types, properties and parameters and to greatly enhance JMS administration.

WebSphere MQ Explorer

The Eclipse based graphical administration tooling, MQ Explorer, introduced in WebSphere MQ V6.0, is further updated in WebSphere MQ V7.0. MQ Explorer enables remote configuration of WebSphere MQ from Linux® x86 and Windows machines. It does not require a local server or client and can be installed on machines without a license.

The main MQ Explorer enhancements are related to:

- General GUI enhancements.
- Browsing messages.
- Mapping between MQ objects and JMS objects.
- Remote queue managers administration.
- Security.
- Queue manager sets.

Several enhancements are fully compatible with WebSphere MQ V6.0 so MQ administrators can benefit from the enhancements to administer V6.0 queue managers.

Working with new properties and parameters

The WebSphere MQ Explorer, MQSC, PCF and control commands have been enhanced to support new MQ object properties and parameters. The changes are mainly related to queue managers, queues, topics, subscriptions, channels and client connections.

Java and JMS related administration enhancements

WebSphere MQ integrates JMS configuration into its graphical, Eclipse based tooling, MQ Explorer, making it easier to design and deploy JMS solutions. JMS objects like connection factories and destinations now appear in the MQ Explorer alongside queues. Since MQ Explorer can remotely configure the entire WebSphere MQ network this makes it easier to explore and configure JMS messaging across the network.

WebSphere MQ V7.0 offers these new administration capabilities for application developers:

- Embedded PCF support for Java.
- WebSphere MQ classes for JMS has been enhanced to provide higher level of serviceability.

Journal time stamps on i5/OS

WebSphere MQ V7.0 for i5/OS® uses a different time stamping technique from earlier versions of WebSphere MQ for iSeries®. This avoids the problem of duplicate time stamps for journals at the end of daylight saving time. There is no longer any need to take any special action after setting the clock back.

3.2.6 Managing Publish/Subscribe

Publish/Subscribe is now fully integrated into the MQ Explorer graphical tooling. Topics can now be administered directly like other MQ Explorer objects such as queues and channels, simplifying administration and security management. Topics can be created using graphical wizards that can also generate corresponding Java Message Service (JMS) Topics. Testing Pub/Sub is now even easier, with built-in tools to send test publications and receive test subscriptions.

The MQSC commands can also be used for manipulating with topic objects and control command **`setmqaut`** can be used for topic object authority settings.

Managing topics

Topics can be managed using MQ Explorer or with MQSC commands. Is it possible to create, alter, display, display status or delete topic objects. JMS topics can be also created and managed using MQ Explorer.

The queue alias type which refers to a topic can be created and managed using MQ Explorer or with MQSC commands.

Appropriate authority settings for topic objects can be set using MQ Explorer or with the `setmqaut` control command.

Managing subscriptions

Subscriptions can be managed using MQ Explorer or with MQSC commands. Is it possible to create, alter, display, clear, display status or delete subscriptions.

3.2.7 z/OS enhancements

The most important enhancement in WebSphere MQ for z/OS is Publish/Subscribe support, which is totally new on this platform. There are also other new features as described below.

New Publish/Subscribe for z/OS

Pub/Sub was introduced in WebSphere MQ in V5.3 for distributed platforms. WebSphere MQ V7.0 provides a fully functioning native Publish/Subscribe feature for both z/OS and distributed platforms. It was previously only available in WebSphere Message Broker.

Mixed case profile management

In WebSphere MQ V7.0 there are new RACF® classes to provide support for mixed case security profiles and topic object security in WebSphere MQ.

To support the mixed case profiles a new queue manager a parameter called SCYCASE (Security Profile Case) has been introduced. There are changes to the “REFRESH SECURITY” MQSC command, to refresh the new MQ RACF classes.

Using WebSphere MQ Explorer without CAF

MQ Explorer can be used to remotely administer and monitor MQ objects, including topics and other Pub/Sub facilities, on z/OS queue managers.

WebSphere MQ V7.0 for z/OS introduces a limited capability to allow MQ Explorer to administer z/OS queue managers at this version without purchasing a license for the Client Attach Facility (CAF). A license still needs to be purchased to allow any other type of WebSphere MQ Client application to connect to the queue manager.

This makes available the great benefits of using the features and graphical user interface of MQ Explorer to administer z/OS queue managers which did not previously have this license.

WebSphere MQ for z/OS listener

MQ Explorer now supports the starting and stopping of the TCP/IP listener in the channel initiator on WebSphere MQ V7.0 for z/OS. This was not possible in prior versions of MQ Explorer and WebSphere MQ for z/OS.

CICS OTE

The CICS Open Transaction Environment (OTE) allows transactions to run under their own TCBs rather than all running on the Quasi-Reentrant (QR) TCB which is normally used.

The benefits are:

- No external change for applications.
- Exits (data conversion, API crossing) need to be thread safe. If not declared thread safe, WebSphere MQ reverts to previous behavior.
- More efficient use of TCBs, especially when mixing calls to WebSphere MQ and DB2.

3.3 Positioning in WebSphere product family

Note to Reviewer: This section should be reviewed and may be changed by SWG to meet their requirements and visions for production positioning.

WebSphere MQ forms the key messaging integration layer of the IBM WebSphere software platform, helping you to attain your business goals.

A message queuing infrastructure built on WebSphere MQ technology provides an available, reliable, scalable, secure and maintainable transport for messages with assured once-only delivery for your business applications and with many other products in the WebSphere family.

3.3.1 Enhanced Enterprise Service Bus

Together with IBM WebSphere Application Server and IBM WebSphere Message Broker, WebSphere MQ provides an ideal basis for implementing your Enterprise Service Bus (ESB). An ESB enables software applications running on different platforms, written in different programming languages and using different messaging models to communicate with each other, without requiring expensive, time-consuming reengineering.

Note: To learn more about how an ESB can help you integrate the diverse elements of your IT environment, refer to IBM web page:

`http://www.ibm.com/software/integration/esb`

The WebSphere product family can use WebSphere MQ ESB implementation as transport layer for application data and interoperability logic.

For example, WebSphere Application Server and WebSphere Process Server are typical products which can benefit from WebSphere MQ capabilities or ESB implementation.

WebSphere MQ and WebSphere Application Server

Java 2 platform Enterprise Edition (J2EE) compliant application servers, such as WebSphere Application Server (WAS), provide a framework within which applications can be developed and hosted.

WebSphere Application Server V6 is supplied with an embedded provider for JMS functionality called WebSphere Platform Messaging. WebSphere Platform Messaging is a separate technology to WebSphere MQ, and provides point to point and Publish/Subscribe message queueing functionality.

WebSphere MQ can be configured as a JMS provider for WebSphere Application Server V6.0. WebSphere Platform Messaging infrastructures can also be interconnected with WebSphere MQ infrastructures.

WebSphere MQ incorporated with WebSphere Message Broker and WebSphere Platform Messaging network is also called WebSphere Enhanced ESB.

3.3.2 Foundation for SOA

In a Service-oriented architecture (SOA), an integration layer, often referred to as an Enterprise Service Bus (ESB), enables and optimizes information distribution between an organization’s service components. Underpinning the ESB layer is a messaging backbone that provides the transport to move data around the organization. As a key member of the WebSphere software portfolio, WebSphere MQ delivers the messaging backbone that can help you take the first step to SOA.

WebSphere MQ enables Simple Object Access Protocol (SOAP) interactions to flow over its messaging backbone between Web service requesters and providers. Legacy and batch applications that are Web services-enabled can also benefit from using WebSphere MQ in its asynchronous mode as a buffering mechanism to regulate the flow of requests made to these systems. It makes an ideal transport for adding reliability and traceability to services connecting to the ESB, supporting a SOA.

Note: To learn more about how an ESB can help you integrate the diverse elements of your IT environment, refer to IBM web page:

`http://www.ibm.com/soa`

Part 2

WebSphere MQ V7.0 enhancements and changes

This part explains the new features and enhancements in WebSphere MQ V7.0 in detail, describes the installation tasks and the migration path from the previous supported versions. The main new features and enhancements covered are:

- Integrated Publish/Subscribe engine
- WebSphere MQ Client enhancements including read ahead, conversation sharing and asynchronous put
- New MQI functions provide Pub/Sub, Call-back, Selectors and Message Property capabilites
- Improved JMS MQ integration
- Administration enhancements including new MQSC commands and MQ Explorer views
- z/OS enhancements including Publish/Subscribe capability and security

This part consists of:

4

Publish/Subscribe integration

In WebSphere MQ V7.0, the Publish Subscribe functionality is integrated into the WebSphere MQ queue manager. This chapter starts with a brief introduction to concepts of WebSphere MQ Publish/Subscribe and then gives a conceptual overview of the Publish/Subscribe engine in v7.

This chapter covers the following topics:

4.1 Publishing and Subscribing in WebSphere MQ

Applications that can provide information on a particular subject are referred to as publishers. Applications that intend to consume this information are referred to as subscribers. Publishing applications supply information about a subject, which in turn maps to a destination managed by WebSphere MQ Publish Subscribe (Pub/Sub), also referred to as a topic. Subscribing applications may register their intention to receive information from a particular topic with WebSphere MQ Pub/Sub. Multiple publishing applications can then send information about the subject as WebSphere MQ messages to the topic managed by WebSphere MQ Pub/Sub. Distribution of this information to registered subscribers is then the responsibility of WebSphere MQ Pub/Sub. In essence, WebSphere MQ Pub/Sub decouples publishing applications from subscribing applications. This decoupling of publishers and subscribers can allow for greater scalability and a more dynamic network topology.

4.2 WebSphere MQ Publish Subscribe in V7.0

In WebSphere MQ V6, a broker that was external to the WebSphere MQ queue manager managed the Publish/Subscribe functionality. One broker could be created for each queue manager and used the same name as the queue manager. This broker used WebSphere MQ facilities to manage interactions between publishing and subscribing applications. This broker is now deprecated and would no longer be supported in WebSphere MQ V7.0. Version 7.0 provides a new Publish/Subscribe engine, which is integrated into the queue manager. This is a major V7.0 enhancement because the WebSphere MQ queue manager now indigenously manages the Publish/Subscribe functionality. The queue manager receives messages from publishers and subscription requests from subscribers for a range of topics. It is then responsible for routing these messages to the target subscribers. Chapter 11, “Installation and migration” on page 227 presents a guide to making the WebSphere MQ V7.0 and V6 Publish/Subscribe engines interoperable. Upcoming sections of this chapter discuss major Publish/Subscribe enhancements in the WebSphere MQ V7.0 product.

The fact that the Publish/Subscribe functionality is now integrated into the queue manager simplifies MQI application programming a great deal. Until V6 Publish/Subscribe was not a part of MQI and MQI based Publish/Subscribe applications were required to talk to a queued interface to a separate broker process running outside the queue manager. New MQI API calls have now been introduced in WebSphere MQ V7.0 to support the Publish/Subscribe

Requirement. Details of these new API calls are provided in Chapter 6, “Message Queue Interface extensions” on page 87.

Further detail on the types of publications and subscriptions available in WebSphere MQ V7.0 can also be found in Chapter 6, “Message Queue Interface extensions” on page 87.

4.2.1 Topics in WebSphere MQ V7.0

A topic refers to the subject on which publishers provide information. Subscribers interested in information on this topic can either subscribe to a topic object or a topic string to receive these publications.

4.2.2 Topic strings and Topic objects

The topic string is the central concept in WebSphere MQ Publish/Subscribe because it connects publishers and subscribers together. Publishers can now publish to a topic string and subscribers can subscribe to these publications using this topic string. The topic string can span up to 10,000 characters. A new data-type called as the Variable Length String has been introduced in WebSphere MQ in order to support this requirement. The structure and semantics of the topic string is controlled by the slash (/). For example, there can be a high level topic called DELI, which might be divided into separate sub-topics relating to different categories of products that the DELI sells, such as:

Example 4-1 `Topic String Examples`

```
deli/fresh/
deli/fresh/fruit
deli/tinned/nuts
```

Usage of a topic string supports wild cards. Subscribers can use wild cards hash (#) and plus (+) to subscribe to a range of topics. Both hash (#) and plus (+) support topic level substitution. The hash can substitute for multiple levels in the topic hierarchy, whereas the plus can substitute for a single level in the topic hierarchy. It is recommended not to use these characters in your topic strings when publishing. Further detail on how to achieve this can be found in Chapter 9, “Publish/Subscribe management” on page 193.

Note: For compatibility with previous versions, an alternative wildcard scheme can be used. Wildcards question-mark (?) and asterisk (*) then stand for character substitution in place of the topic level substitution as supported by the hash (#) and the plus (+). (Refer to infocenter).

Topic strings imply a sense of hierarchy in the topic structure. This hierarchy can be represented as a topic tree, which is an internal representation for this hierarchy. The topic tree has a root node, which is `SYSTEM.BASE.TOPIC`.

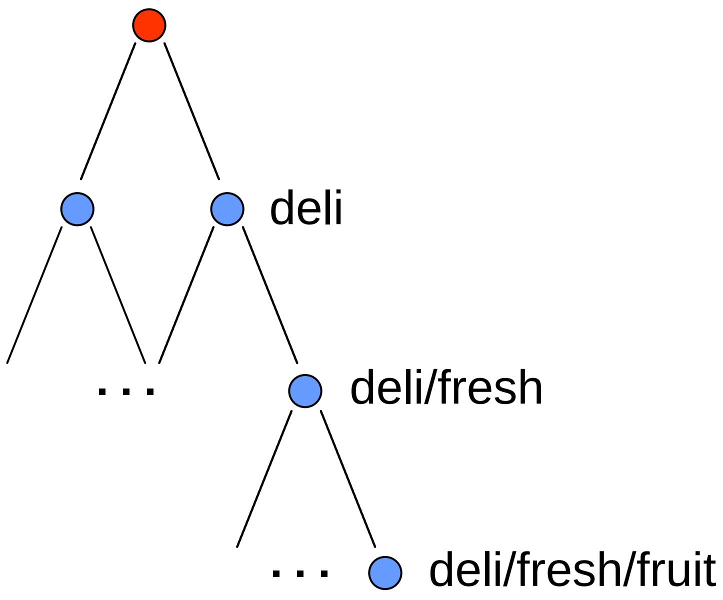


Figure 4-1 Topic Tree

> **Note:** A subscription made to the topic string '#' would normally send publications from all topics in the topic tree to the subscriber, however should the administrator wish to partition the tree such that a wildcard subscription does not match part of the tree then he can define the topic object with the attribute WILDCARD (BLOCK) which prevents a wildcard from processing that part of the tree

Topic objects are administrative objects, which can be defined in WebSphere MQ. Administrative topic objects are a required in order to be able to define attributes for certain portions of the topic tree for instance, to set up authority checking on specific topics. For example, authority checking may include whether that portion of the tree can be published or subscribed to. Topic objects other than the pre-defined base topic (the root of the topic tree), are not required if security and topic attributes are the same for the whole topic tree, since these settings can be inherited from the pre-defined base topic. Thus, a user could get a publish/subscribe application up and running without administratively defining any topic objects at all.

Topic strings are used to match information from a publisher to a subscriber interested in that information. Topic strings do not have to be pre-defined, but come into existence on the fly when subscribing and publishing applications use them. Consider a publisher application publishing to a topic string called deli/fresh/fruit. Note that no administrative topic objects get created at this time. Rather, the nodes on the corresponding tree (see figure 1) are referred to as non-administrative topics. You may choose to define an administrative object for any node on this sub-tree (say, /fresh/fruit) only if there is a need for associating specific, non-default attributes with that particular node.

Further, defining an administrative object on /fresh/fruit may not be necessary if there is an administrative object defined for deli/fresh, which already has these specific non-default attributes defined as in this case, the node /fresh/fruit inherits these attributes. Also, there is no one-to-one mapping between the administrative topic objects and the nodes in the topic tree.

Defining Topic Objects bears no impact on subscribing applications. Although, subscribers can still use topic strings to subscribe to topics, they can also choose to use a topic object name for subscribing to that topic. WebSphere MQ Publish Subscribe allows administrators to shield application developers from portions of the topic tree by allowing them to define an administrative topic at the highest point in the topic tree up till which the developers need to know and then let them create non-administrative nodes below that point by using topic strings. For example, if we had an administrative object called DELI.FRESH defined for deli/fresh already, then publishers and subscribers can use this object name in conjunction with the name of the product type they are interested in as the topic string say, Fruit. This has the same effect as publishing or subscribing to deli/fresh/fruit.Further detail on how to achieve this can be found in Chapter 9, “Publish/Subscribe management” on page 193.

4.2.3 Topic Alias

An alias queue is a WebSphere MQ object that you can use to access another queue. The queue resulting from the resolution of an alias name (known as the base queue) can be a local queue, the local definition of a remote queue, or a shared queue (a type of local queue only available on WebSphere MQ for z/OS). It can also be either a predefined queue or a dynamic queue, as supported by the platform.

WebSphere MQ V7.0 introduces an extension to the `ALIAS QUEUE` object which allows an Alias queue to be mapped to a topic object. Typically this is useful for migrating point-to-point applications to a publish/subscribe message model. This is possible because you can define an alias queue object on the queue name being used by the message producer and consumer applications. Further, now with WebSphere MQ V7.0, you can map this queue alias to a topic object. This means that the message producer application is now capable of publishing messages to this topic even if it would be actually putting messages using the same queue name. On the message consumer's side, we need to define a new local queue, for the message consumer to GET messages from. This means that the message consumer application has to be modified to GET messages from this new queue. This is because in a Publish/Subscribe environment, you need to define a Non-managed subscription for the WebSphere MQ V7.0 Publish/Subscribe Engine to deliver subscriptions to the queue from which the message consumer application is GETing messages. A Non-managed subscription is a way for the Subscriber application to specify a destination to which it wants its publications delivered.

Conceptually, the introduction of the topic alias also brings in the benefits of the Publish/Subscribe Topology. For example, from an administration perspective, consider a queue to which statistics messages are written. The point to point paradigm allows for a single message producer to put messages to this queue which are then consumed by a single message consumer. By defining a Queue Alias which points to a topic object, it is possible for each application interested in processing statistics messages to subscribe to the topic, which allows multiple consumers to consume the statistics information. Further detail on how to map an alias queue object to a topic object is provided in Chapter 9, "Publish/Subscribe management" on page 193.

4.2.4 Topic Security

Nodes in the topic hierarchy which have a topic object associated with them are known as Admin Nodes. The delegation model of inheriting attributes from the nearest admin node in the topic hierarchy, as discussed in previous sections, holds true for security as well. Nodes which are automatically generated inherit the properties of the nearest parent Admin Node in the topic hierarchy. Please note, that once an application is permitted to publish or subscribe at a parent level admin node, it cannot be denied publishing or subscribing authorities on a child level admin node.

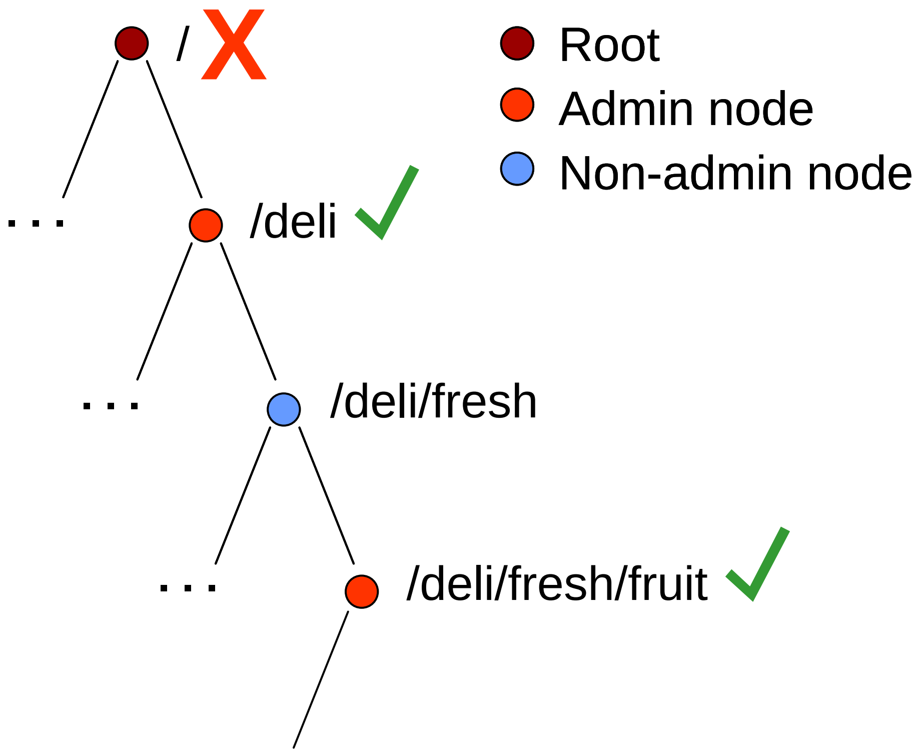


Figure 4-2 Delegation of Topic Security Attributes

Suppose, if a given application wants to subscribe to the topic string /deli/fresh/fruit and if /deli allows this application to subscribe, access would be granted to this application to subscribe to /deli/fresh/fruit. Publish/Subscribe Security can be managed in WebSphere MQ V7.0 using commands as well as the MQ explorer. Further detail on how to achieve this can be found in Managing the Pub/Sub environment.

4.3 Selectors

Message Selectors allow an application to specify the messages that it is interested in by using message properties. Only messages whose properties match the selector are delivered to the application. Please note that selection may only be performed on the properties associated with a message, not on the message payload itself. Even if a message contains no message properties (other than header properties) then it may still be eligible for selection.

Further, WebSphere MQ v7 now provides native support for Message Properties and allow users to set and get a user property in a message, whether using JMS or using MQ API, with the same property name. Further detail on using Message Properties in MQ can be found under Message Properties and Handles in Chapter 6, “Message Queue Interface extensions” on page 87.

Users of WebSphere MQ V6 note that the V6 implementation required that message selection be performed on the client side. With this implementation, messages are browsed sequentially from a queue until the client finds one that matches the selection criteria, at which point a destructive MQGET is performed on that message. The WebSphere MQ V7.0 makes message selection much more efficient by introducing Selectors on the queue manager. This means that the queue manager uses a selector to find messages that match the selection criteria specified by the client. These messages are then sent to the client. In essence the selection functionality is now moved from the client to the server.

4.3.1 Syntax for WebSphere MQ V7.0 Selector Strings

An MQ message selector is a String, whose syntax is based on a subset of the SQL92 conditional expression syntax. The order in which a message selector is evaluated is from left to right within a precedence level. You can use parentheses to change this order.

Literals: string literals in a selector string are specified using a single quote for example, ‘`MQ`’ or ‘`WebSphere MQ`’. Byte literals, which are one or more pairs of hex characters, need to be specified in double quotes for example, `0XD43A`. Boolean literals can take values TRUE and FALSE. All kinds of numeric literals, which include decimal, hex and octal numbers do not need to be specified in quotes.

Identifiers: An identifier is a variable-length character sequence that must begin with a valid identifier start character, followed by zero or more valid identifier part characters. Identifiers are either header field references or property references and are case-sensitive. For example, in the selector string `COLOR IS RED`, COLOR is an identifier and RED is a literal.

Operators: There are three kinds of operators that you can use in a selector string i.e. Arithmetic Operators, Logical Operators and Comparison Operators.

Valid arithmetic operators include + (both unary and binary plus), - (both unary and binary plus), * (multiplication) and / (division). The precedence order followed for these operators is +, -, *, / in that order.Valid Comparison Operators include = (equal to), > (greater than), >= (greater or equal to), < (less than), <= (less than or equal to), <> (not equal). String and Boolean comparison is restricted to = and <>. Two strings are equal only if they contain the same sequence of characters.Valid Logical operators include AND, NOT and OR. There is a precedence order for logical operators which is: NOT, AND, OR in that order. Besides these, there are two valid operators: LIKE, which is used for pattern matching and BETWEEN which is used in arithmetic comparisons.

Also, note that traditional type conversion rules do not hold true for selectors. For example, if your Message had a Message Property as a string value and then use a selector to query it as a numeric value, the expression returns FALSE. The only exception to this rule is that it is valid to compare exact numeric values and approximate numeric values. In general, if there is an attempt to compare different types, the selector is always false.

Note: If an identifier references a message property, which does not exist, then this property is assumed to have the value of `NULL` or `Unknown` and so such a message may still satisfy a selection string like `COLOR IS NULL`, where `Color` does not exist as a message property in the message.

Managing JMS Header References: JMS field and property names which map to property names or MQMD field names may be used as valid identifiers in a selection string. WMQ maps the recognized JMS field and property names to the appropriate message property. For instance, the selection string "JMSPriority >= 0" selects on the Priority property found in the jms folder of the current message. JMS Message header field references in selectors are restricted to: JMSDeliveryMode, JMSPriority, JMSMessageID, JMSTimestamp, JMSCorrelationID, JMSType, JMSMessageID, JMSTimestamp, JMSCorrelationID, and JMSType values can be null, and if so, are treated as a NULL value.

Any name beginning with JMSX or JMS_ is a JMS-defined property name and is mapped to the appropriate MQMD field or property according to the rules used by the WMQ JMS implementation. Rules governing this mapping are discussed in Mapping JMS messages onto WebSphere MQ messages under Using Java on the WebSphere MQ V7.0 Infocenter. Any name that does not begin with JMS is an application-specific property name. If there is a reference to a property that does not exist in a message, its value is NULL. If it does exist, its value is the corresponding property value. Any JMS property names not matching the recognized set of header fields or JMSX/JMS_ property names are assumed to be user property names requiring no mapping. When used in a message selector JMSDeliveryMode is treated as having the values `PERSISTENT` and `NON_PERSISTENT`. This means, for example, that the selection string `JMSDeliveryMode = PERSISTENT` is valid, whereas `JMSDeliveryMode = 1` is not.

Table 4-1 Selector string examples

Type	Selector	Evaluates to	Governing Rules
Selectors using ByteLiterals Assume myBytes = 0AFC23 is the Message Property.	myBytes = “0x0AFC23”	TRUE	► Matching a selector byte string to a message property of type MQTYPE_BYTE_STRING is performed without any special action taken on leading/trailing nulls, that is, they are treated as just another character. ► Whether your machine is Big Endian or Small Endian does not matter here ► The length of both selector and Message Property byte strings should therefore be equal and the sequence of bytes should be exactly the same.
	myBytes = “0xAFC23”	MQRC_SELECTOR_SYNTAX_ERROR. Reason: The numbers of bytes are not a multiple of two.	
	myBytes = “0x0AFC2300”	FALSE Reason: The message property is of type MQTYPE_BYTE_STRING, the trailing null character in the Selector is treated like a normal character, and thus matching evaluates to FALSE.	
	myBytes = “0x23FC0A”	FALSE Reason: Big Endean or Small Endean does not matter.	

Type	Selector	Evaluates to	Governing Rules
Selectors using Exact and Approximate Numeric Literals	NoItemsInStock > 20. Assume NoItemsInStock =22	TRUE	► An exact numeric literal is a numeric value without a decimal point, such as 57, -957, +62; numbers in the range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 are supported and are internally stored as signed long values ► Hex number, e.g. 0xA, 0xAF, 0x2020. Hex numbers should begin with a zero, followed by an upper/lowercase 'x'. The remainder of the literal should contain n valid hex characters where n > 1 ► Octal number, e.g. 0177, 0713. Octal numbers begin with a leading zero which is followed by n digits in the range 0-7, where n >= 1. It should be noted that a leading zero which is followed by one or more digits is always interpreted as being the start of an octal number which means that it is not possible to represent a zero-prefixed decimal number in this way, e.g. '09' would return a syntax error as 9 is not a valid octal digit. ► Exact numeric literals may contain a trailing upper/lowercase 'L' character. This does not affect how the number is stored or interpreted but is accepted as it is a valid addition to a numeric literal.
	(Temperature > -10L) AND (Temperature < 50). Assume Temperature =-12	FALSE	
	AnOctal + AnOtherOctal <> 0188. Assume the total comes to a number other than 136, which is the decimal representation of 0188	TRUE	
	Inches * +2.54e-2 > 1.0+2.54e-2 evaluates to 0.0254. Assume Inches to be 1.0.	TRUE	

Type	Selector	Evaluates to	Governing Rules
Selectors using Logical Operators and Pattern Matching	Country NOT IN ('UK', 'US', 'France')	Is FALSE for 'UK' and TRUE for 'Peru'	► Comparison or arithmetic with an unknown value always yields an unknown value. ► If the identifier of an IN or NOT IN operation is NULL, the value of the operation is unknown. ► The IS NULL and IS NOT NULL operators convert an unknown value into the respective TRUE and FALSE values ► If the identifier of a LIKE or NOT LIKE operation is NULL, the value of the operation is unknown. ► '_' Stands for any single character and '%' stands for any sequence of characters (including the empty sequence)
	JMSType = 'car' AND color = 'blue' OR weight > 2500Note: Demonstrates the use of a JMS Header Reference.	Because AND is higher in precedence than OR, the result of JMSType='car' AND color='blue' is then ORed with the result of weight > 2500	
	PHONE LIKE '12%3'	TRUE for 123 and 12993 and FALSE for 1234	
	GAMEDECISION LIKE 'L_SE'	TRUE for LOSE FALSE and false for LOOSE	

4.4 Distributed Publish/Subscribe

Distributed Pub/Sub helps applications connected to separate queue managers in communicating via pub/sub. Distributed Pub/Sub allows for two types of topologies, which are Hierarchical and Pub/Sub Clusters (also known as Collectives).

4.4.1 Publish/Subscribe Clusters

A publish/subscribe cluster uses WebSphere MQ queue manager clustering for queue manager connectivity. In a queue manager cluster, all queue managers are interconnected with each other. What makes this queue manager cluster a Publish/Subscribe cluster is the definition of a clustered topic within the cluster. Although the clustered topic is created on one queue manager, the definition is shared with all queue managers in the cluster. It is at this time of defining the clustered topic when the queue managers in the cluster get notified of each other. This mandates that all queue managers in the cluster be fully interconnected with each other. Also, all publications that come to the clustered topic are sent to all the queue managers in the cluster and are thus sent to all subscribing applications connecting to each queue manager in the cluster.

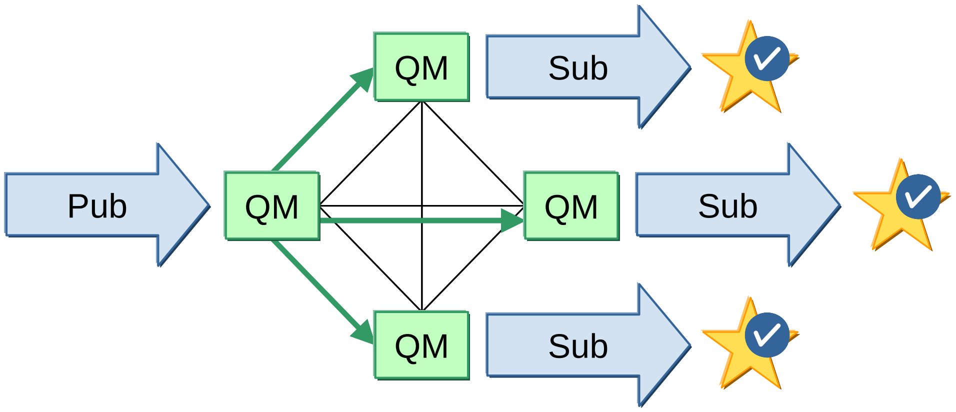


Figure 4-3 Publish/Subscribe Clusters

Subscribing to Clustered Topics: When an application subscribes to a topic that resolves to a clustered topic, WebSphere MQ creates a proxy subscription and sends it, from the queue manager to which the subscriber connected, to all other queue managers of the cluster in which the clustered topic object is defined. Proxy subscription forwarding is a routing mechanism in which subscriptions are only forwarded to directly connected queue managers. In the case of Publish/Subscribe clusters, as all queue managers are interconnected, subscriptions are forwarded to all members of the Publish/Subscribe Cluster. This also holds true if the queue manager on which the clustered topic object was defined becomes unavailable, which means that subscriptions coming to the other queue managers in the cluster are still sent out as proxy subscriptions to all other queue managers in the cluster. This is possible because the clustered topic object definitions are available on other queue managers also.

Publishing to Clustered Topics: Publications in a Publish/Subscribe cluster also follow the Proxy Subscription routing mechanism. This means that a publication to the clustered topic on a queue manager, which is part of the cluster, is sent out to all directly connected queue managers that either have a local subscription or have received a proxy subscription.

Note: A single queue manager can be a member of more than one Pub/Sub cluster. This is done to create a cluster gateway between two clusters, so that messages originating in one Pub/Sub cluster can be routed to another Pub/Sub cluster. Although a WebSphere MQ queue manager can be a member of more than one Pub/Sub cluster, publications are not passed from one cluster to another by means of overlapping clusters. The scope of proxy subscriptions is limited to the single cluster in which the clustered topic is defined. In order to connect two Pub/Sub Clusters, a Hierarchical Distributed Topology can be used.

4.4.2 Hierarchical Distributed queue managers

Queue managers can be connected together using a connection-time parent and child relationship. The Hierarchical Topology uses Proxy Subscriptions as a routing mechanism. Thus, it takes some time for the proxy subscription to propagate around all nodes in the network. The consequence of this is that, once a subscription has been made, remote publications are not necessarily received immediately. An alternative routing mechanism to proxy-subscription forwarding is publish-everywhere which works by publishing to all directly connected queue managers regardless of there being local or proxy-subscriptions on those queue managers.

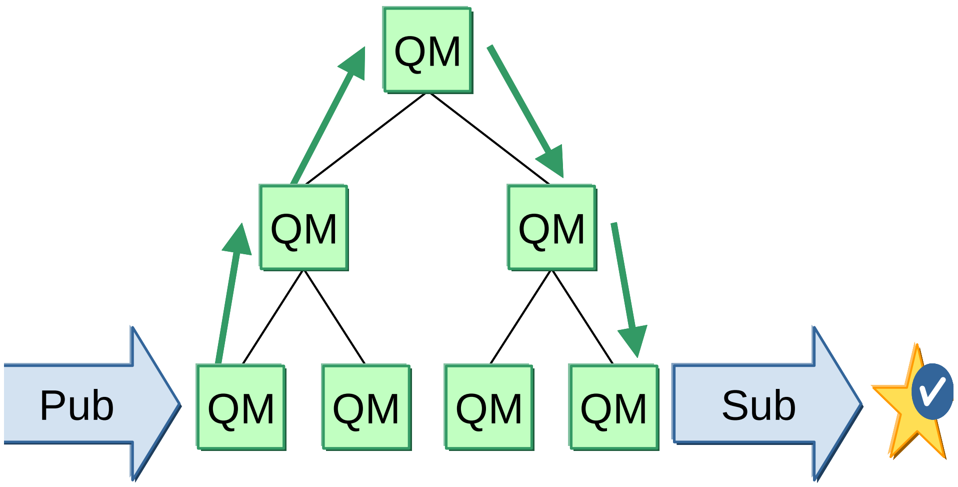


Figure 4-4 Hierarchical Distributed queue managers

Note: If one queue manager is attached by a hierarchical connection or as part of a pub/sub cluster to more than one queue manager with the same queue manager name, this can result in publications failing to reach one or all of the identically named remote queue managers. As with point to point messaging, it is strongly recommended that queue managers have unique names, especially if they are directly or indirectly connected in a WebSphere MQ network.

4.4.3 Loop Detection

In a distributed publish/subscribe network, it is important that publications and proxy subscriptions cannot loop, as this would result in a flooded network with connected subscribers receiving multiple copies of the same original publication.

As publications move around a publish/subscribe topology each queue manager adds a unique fingerprint to the message header. Whenever a publish/subscribe queue manager receives a publication from another publish/subscribe queue manager, the fingerprints held in the message header are checked. If the queue manager's own fingerprint is already present it means that the publication has fully circulated around a loop, so the queue manager discards the message, and adds an entry to the error log.

4.4.4 Scope of Publications and Subscriptions in a Distributed Publish/Subscribe Environment

The scope of a publication or a subscription in a Distributed Pub/Sub Environment is defined as the queue managers in a WebSphere MQ network to which that publication or subscription is delivered or propagated to.

Publication Scope: The scope of publications can be controlled administratively using the PUBSCOPE topic attribute. The attribute can be set to one of the following values:

QMGR: the publication is only delivered to local subscribers.

ALL: The publication is delivered to local subscribers and remote subscribers via directly connected queue managers.

Subscription Scope: Similarly, the scope of subscriptions can be controlled administratively using the SUBSCOPE topic attribute. The attribute can be set to one of the following values:

QMGR: The subscription is not propagated to directly connected queue managers, and only receive publications from local publishers.

ALL: The subscription is propagated directly to connected queue managers, and receive publications from local publishers and remote publishers via directly connected queue managers

Note: In the case of a Pub/Sub Cluster, on defining the PUBSCOPE or SUBSCOPE with the QMGR attribute the scope of publications/subscriptions based on that topic becomes local on. However, the definition of the clustered topic object is still shared with other queue managers of the cluster.

5

WebSphere MQ Client enhancements

This chapter describes the new features and enhancements in WebSphere MQ V7.0 which apply to the WebSphere MQ Client. It also covers relevant changes to other components of MQ which use the client.

MQ Client is a set of libraries or classes which have a light-weight software footprint yet provide full programmatic access to all the Message Queue Interface (MQI) calls. It does not require a queue manager to reside on the same system as the program. By utilizing a communications protocol, such as Transmission Control Protocol/Internet Protocol (TCP/IP), the client libraries communicate with a queue manager via a MQI Channel. MQ Client is available for Windows and Unix platforms and it can connect to any platform which can run a WebSphere MQ queue manager.

Refer to the manual *WebSphere MQ Clients* SCXX-XXXX-XX for further details on platform requirements, installation, configuration, administration and programming using the client. WebSphere MQ manuals are available on Internet at the IBM Information Center:

http://to.be/supplied.html

The chapter includes the following topics:

- "Overview of enhancements"

- “Full duplex channels, heart beating and quiesce”
- “Conversation sharing”
- “Read ahead”
- “Asynchronous put”
- “Instance limits on SVRCONN channels”
- “Weighted selection on CLNTCONN channels”
- “Reconnecting via a previously used channel”
- “Maximum message length increased on MQSERVER environment variable”
- “Security exit details in WebSphere MQ Explorer”
- “Using MQ Explorer without a Client Attach Facility (CAF) license on z/OS”
- “Compatibility”

5.1 Overview of enhancements

Many of the enhancements to WebSphere MQ Client reflect the underlying behavior of the redesigned JMS WebSphere MQ integration layer. Three primary changes have been made:

- The TCP/IP protocol which MQ uses over the channel between a client and a queue manager has been converted from half duplex to full duplex. Network failures are detected earlier by allowing independent heartbeats to be performed in each direction on the channel. Channel stop requests from the queue manager can now be processed immediately.
- Multiple connections established by threads of a client program can share one instance of a TCP/IP client channel rather than running their own instances. This effectively reduces the number of running channels on the queue manager and therefore makes more efficient use of resources.
- The “read ahead” and “asynchronous put” features provide an increase in throughput for getting and putting messages under specific circumstances where lower Quality of Service (QoS) can be tolerated on TCP/IP client channels.

Other new features improve the operational management, programming and administration aspects of the client:

- New channel attributes allow limits to be imposed at two levels on the number of instances of running client channels. This prevents a client or JMS MQ program from running the maximum number of channel instances available on a queue manager and hence denying other client programs from connecting and all other types of MQ channels from starting.
- A client program can request connection to a queue manager name which is prefixed by an asterisk character (*). This directs MQ to attempt connection using an alphabetic list of CLNTCONN type channels defined a Client Channel Definition Table (CCDT) file. New parameters have been added to the CLNTCONN channel definition to allow random selection based on a relative weighting and availability of queue managers.
- A client program can now obtain the name of the channel which was selected by a successful call to MQCONNX. The channel name can also be passed to MQCONNX to override the normal algorithm which is used to select a channel in a CCDT file. This allows programs to connect to a previously used channel and queue manager without knowing the exact name.
- On a CLNTCONN channel defined using the MQSERVER environment variable, the maximum message length has increased from 4MB to 100MB.

There are changes to other components of WebSphere MQ V7.0 related to the client:

- Security Exits can now be directly enabled and configured in MQ Explorer for Client channel connections to remote queue managers. Previously this could only be done using a Client Channel Definition Table (CCDT) file.
- MQ Explorer can now be used to administer z/OS queue managers without purchasing a license for the Client Attach Facility (CAF).

The client also supports other new features of WebSphere MQ V7.0 which have been incorporated into the general MQI. They are described in detail in Chapter 6, “Message Queue Interface extensions” on page 87:

- Publish/Subscribe: The principles are explained in Chapter 4, “Publish/Subscribe integration” on page 45.
- Getting and setting Message Properties using Message Handles.
- Message Selectors.
- Cooperative browsing using Message Tokens.
- Asynchronous Consume and Event notification using Call-back.

The following sections contain detailed descriptions of enhancements which are targeted specifically at WebSphere MQ Client. There is discussion on how they can best be used to improve your existing environment, or design a new application to use the WebSphere MQ V7.0 client features efficiently. This includes figures and short programming examples in C where appropriate.

Refer to Scenario: News using Client for details of the “News” component which illustrates WebSphere MQ Client programs making appropriate use of read ahead and asynchronous put. Other components also use the client but do not necessarily take advantage of the enhancements in WebSphere MQ V7.0.

5.2 Full duplex channels, heart beating and quiesce

The communications session between a WebSphere MQ Client program and the queue manager is defined using a CLNTCONN type channel on the client and a SVRCONN type channel on the queue manager. They have identical names.

WebSphere MQ V7.0 uses a full duplex protocol when the transport type is specified on the both types of channel as TCP/IP and the conversation sharing parameter SHARECNV is greater than zero on both channels. Conversation sharing is a new feature and is described in the next section.

Full duplex means that information can be sent from either end of the session at any time. Heart beats are short flows of information which are sent at a regular interval. Their only purpose is to confirm that the session is still active. Heart beats can now be performed from both the client end and the queue manager end at independent rates, as specified by the HBINT parameter on the channels. This leads to earlier detection of communications network failures and other channel problems. The client program and the queue manager can now carry out recovery and reconnecting functions in a more timely manner, resulting in an overall improvement to the Quality of Service.

Previous versions of MQ use a half duplex protocol, where a heart beat could only be performed from the queue manager end during a MQGET operation with a WaitInterval specified. This limited its usefulness for detecting problems. Refer to the description of the HBINT parameters in the manual *WebSphere MQ Intercommunication,* SCXX-XXXX-XX for further details of this feature.

The following two figures show the difference between half duplex and full duplex operation of a WebSphere MQ Client channel:

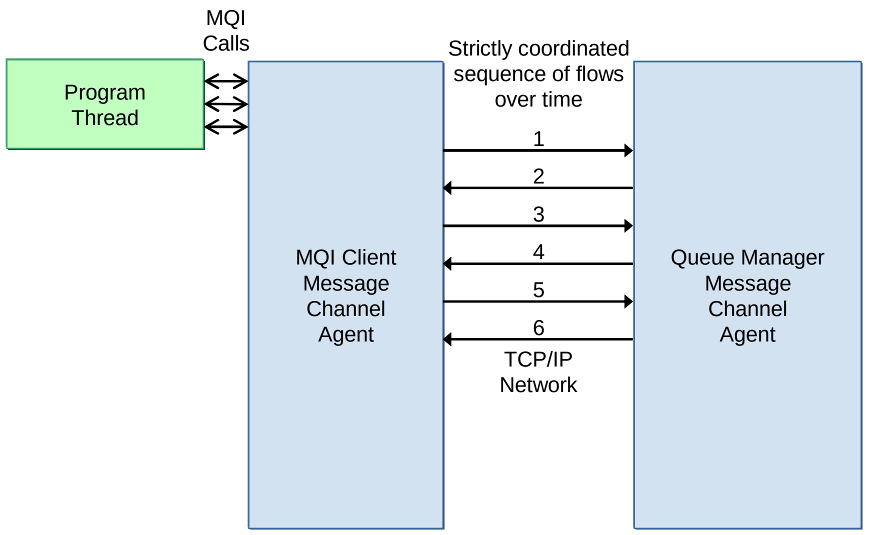

MQI Calls
Program Thread
Strictly coordinated sequence of flows over time
1
2
3
4
5
6
TCP/IP Network
MQI Client Message Channel Agent
Queue Manager Message Channel Agent

Figure 5-1 Half duplex client channel operation in WebSphere MQ

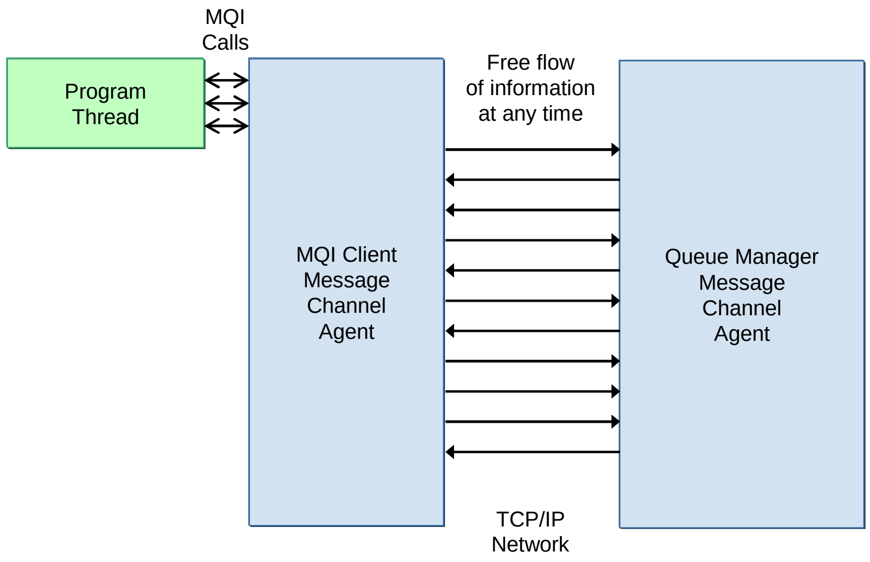

MQI Calls
Program Thread
MQI Client Message Channel Agent
Free flow of information at any time
Queue Manager Message Channel Agent
TCP/IP Network

Figure 5-2 Full duplex client channel operation in WebSphere MQ V7.0

The full duplex protocol also enables the issuing of a STOP CHANNEL(ChannelName) MODE(QUIESCE) command to be immediately communicated to the client. This results in the program being returned the appropriate Reason Code on its current or next MQI call, and the communication session on the channel shuts down cleanly in a more timely manner.

5.3 Conversation sharing

WebSphere MQ V7.0 introduces the ability for many threads in one client program to connect to the same queue manager and only use one running instance of a client channel. The client protocol conversations for the MQI calls made by each thread are transparently multiplexed over a single TCP/IP session. There is also only one pair of heart beat flows and one flow to stop the channel.

Compared to previous versions of WebSphere MQ where multi-threading client programs could only use multiple channels instances, the conversation sharing feature in WebSphere MQ V7.0 reduces the number of running channels on the queue manager, resulting in more efficient use of resources.

Figures 5-3 depicts many threads within a client program which share a single channel instance to communicate with the queue manager. The JMS MQ provider uses the same methodology.

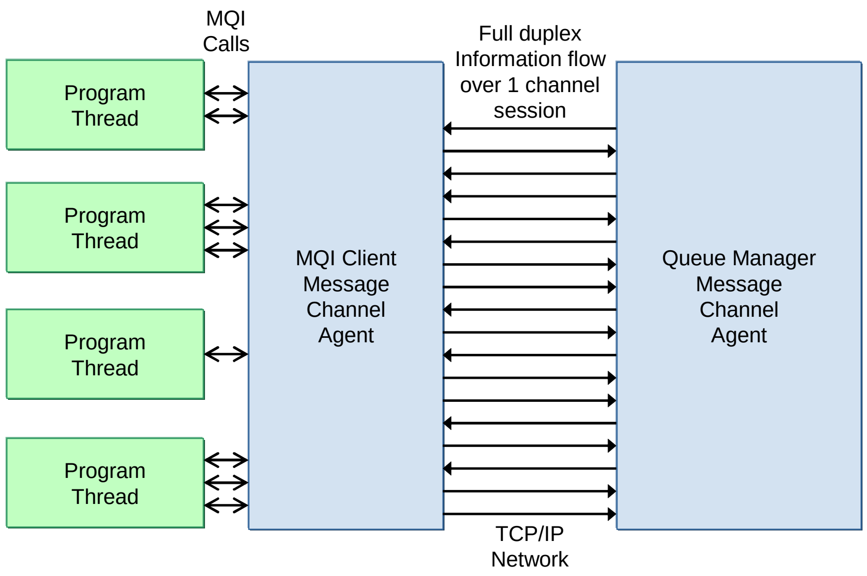

MQI Calls
Full duplex Information flow over 1 channel session
Program Thread
Program Thread
Program Thread
Program Thread
MQI Client Message Channel Agent
Queue Manager Message Channel Agent
TCP/IP Network

Figure 5-3 Multi-thread conversation sharing in WebSphere MQ V7.0

As all threads are sharing the same communications link there is contention for its use. For example, an MQOPEN call in one thread may take longer than expected to execute because many other threads are calling MQPUT and MQGET to process messages.

5.3.1 SHARECNV parameter and management of channel definitions

Conversation sharing is controlled by value of the SHARECNV parameter on both SVRCONN type channels and CLNTCONN type channels. If it is set to 0 in the channel definition on either end of a running pair of channels the client reverts to the behavior of previous versions of WebSphere MQ, where conversation sharing is not available and every thread of the client program has its own running instance of a client channel.

A value of SHARECNV in the range 1 to 999999999 specifies a limit on the maximum number of threads in a single client program which can share a single running channel instance. A program can have more threads than the limit connected to MQ, this results in additional channel instances being started and each runs up to the limit. If there are multiple programs using the same channel name they each have their own limit on the number of threads which can share a single instance.

If the SHARECNV parameter is not set to the same value on either end it uses the lower value.

In the case of CLNTCONN channels defined prior to WebSphere MQ V7.0, the SHARECNV parameter does not exist and conversation sharing is not supported.

When a queue manager is migrated to WebSphere MQ V7.0 it automatically updates all SVRCONN type channel definitions to define all the new parameters. SHARECNV set to 10. This also the default for new channels, but it may have changed if the SHARECNV parameter on the SYSTEM.DEF.SVRCONN channel has been altered since the queue manager was migrated or created.

To enable conversation sharing on a Client Channel Definition Table (CCDT) file which was created prior to WebSphere MQ V7.0, the CLNTCONN channels defined in the file needs to be manually altered using a WebSphere MQ V7.0 queue manager to set the SHARECNV parameter to a non-zero value. Refer to the sections on environment variable MQCHLTAB and the Client Channel Definition Table in the manual *WebSphere MQ Clients,* SCXX-XXXX-XX for further details.

CLNTCONN channels can be defined and configured dynamically by using the MQSERVER environment variable, the MQCONNX API call, and the Java / JMS classes. Under WebSphere MQ V7.0 they all use a default SHARECNV value of 10. The default can be changed using the SharingConversations field of the MQCD data type which can be specified as an option to the MQCONNX call. It can also be specified using the sharingConversations field of the MQEnvironment and MQChannelDefinition classes in Java.

5.3.2 MQCONNX options

The MQCONNX API call can be used by a client program to connect to a queue manager. Its function is similar to the MQCONN API call but it contains an additional argument to specify options for the client channel, SSL and security.

In WebSphere MQ V7.0 there are two new options on MQCONNX:

MQCNO_ALL_CONVS_SHARE is the default. It specifies that the client channel is to use the normal processing for sharing conversations within the client program.

MQCNO_NO_CONV_SHARING specifies that this thread is to have its own instance of the client channel and there is no conversation sharing with any other thread of the client program.

The following example contains code in the C language which sets a conversation sharing options on a call to MQCONNX:

Example 5-1 C code to set a conversation sharing option for MQCONNX

```
#include <cmqc.h>
#include <cmqxc.h> /* For MQCD structure */
...
MQHCONN hConn = MQHC_UNUSABLE_HCONN;
MQCNO ConnOpts = MQCNO_DEFAULT;
MQLONG CompCode, ReasCode;
...
   ConnOpts.Options = MQCNO_NO_CONV_SHARING; /* Disable sharing */
   MQCONNX( “MYQMGR”, ConnOpts, &hConn, &CompCode, &ReasCode );
   if( CompCode == MQCC_OK )
      ...
```

5.3.3 Displaying channel status

The MQ Explorer and MQSC commands have been enhanced to display channel status information related to conversation sharing for active instances of SVRCONN type channels. The relevant status values are as follows:

CURSHCNV is a new field which reports the number of conversations which are currently connected to the queue manager on the channel instance.

MAXSHCNV is a new field which reports the maximum number of conversations which can be shared on the channel instance. This is the lowest value of the SHARECNV parameter on the associated CLNTCONN and SVRCONN channels.

A value of 0 for CURSHCNV and MAXSHCNV indicates that the channel is running in the mode of operation which was used prior to WebSphere MQ V7.0, where conversation sharing is not supported.

MCAUSER is the effective UserId of the channel instance, as normally negotiated by MQ for a client program. If there are multiple program threads sharing the channel instance and they all have the same UserId, it is reported. If any threads have a different UserId, the MCAUSER is reported as an asterisk (*).

MSGS represents the total number of MQI calls made by all client program threads which are sharing the channel instance.

LSTMSGDA and LSTMSGTI represents the date and time of the most recent MQI call made by any thread which is sharing the channel instance.

Refer to Chapter 8, “Administration enhancements” on page 151 for an example of MQ Explorer being used to display channel status.

5.3.4 Channel exits

There are additional considerations when using Channel Exits on client channels which have conversation sharing enabled. An exit sees the information flows for many independent client threads and may need to serialize access to the MQCD data type. Existing exits on CLNTCONN and SVRCONN type channels should be reviewed for possible impact to their correct operation. The issues are discussed in the section “Implications of sharing conversations“ in the manual*WebSphere MQ Intercommunication* SCXX-XXXX-XX available at:

http:

5.4 Read ahead

A new feature of WebSphere MQ V7.0 Client can provide an increase in throughput for a client program which gets a sequence of non-persistent messages. A good example is an enquiry business function which results in a back-end application generating a number of reply messages on a queue which must then be retrieved by the client program.

While the client program is requesting the messages by making repeated calls to MQGET, the MQ client libraries read ahead on the queue and store additional non-persistent messages in memory on the client system. If the requested message is in memory, it is immediately returned to the MQGET.

Read ahead reduces the overall number of interactions on the client channel and the latency for each MQGET call to communicate with the queue manager and wait for the response to come back.

The messages which are streamed into memory on the client system have been destructively removed from the queue manager. They are then lost if the client program terminates or the client system fails before the program gets all the messages. Therefore the read ahead feature is a compromise between increased throughput and a lower Quality of Service (QoS), where assured delivery is not provided. This may be quite acceptable in some circumstances.

If the read ahead function encounters a persistent message it stops reading messages into the client memory until the persistent message has been got by the client program. The amount of memory allocated by read ahead and how it is utilized for storing messages is managed intelligently by MQ Client.

Read ahead is enabled by specifying an option when a queue is opened or by a default parameter on the queue definition. There are some considerations getting messages and closing queues. These are discussed in the following sections.

Read ahead is also supported by the new MQI verbs MQCB and MQCTL, which provide an asynchronous consumption feature using Call-back. Refer to Chapter 6 for details. The JMS MQ client and the new Subscribe feature also support read ahead.

5.4.1 MQOPEN options to specify read ahead

There are three new options on the MQOPEN verb which control the read ahead feature:

MQOO_READ_AHEAD_AS_Q_DEF specifies that read ahead is determined by the setting of the new parameter DEFREADA on the local queue, model queue or alias queue which is being opened. This is the default if none of the options is specified.

> The DEFREADA parameter may have the following values:
>
> NO indicates that read ahead is not to be enabled. This is equivalent to the behavior in versions prior to WebSphere MQ V7.0 where messages are transported from the queue manager at the time they are requested.
>
> YES indicates that read ahead is enabled.
>
> DISABLED indicates that read ahead is not to be enabled, even if the MQOO_READ_AHEAD option is specified by a program which opens the queue.

MQOO_NO_READ_AHEAD specifies that read ahead is not to be enabled. This is equivalent to the behavior in versions prior to WebSphere MQ V7.0.

MQOO_READ_AHEAD specifies that read ahead is to be enabled, unless overridden by the DEFREADA parameter of the queue being set to DISABLED.

Only one of these three parameters can be specified as an option on an MQOPEN. The same options are available on the new MQSUB verb, but they have a prefix of “MQSO_”. Refer to Chapter 6 for details.

5.4.2 MQGET considerations

Messages can be requested by the client program with matching Message Id and/or Correlation Id but this may result in inefficient use of read ahead. The client streams the messages into memory on the client system but they may not be messages which are requested by the program.

The first request to get a message from a queue is determined if is being browsed or got destructively. It cannot be changed after that without reopening the queue.

Many of the options for getting messages result in read ahead being disabled on the queue:

MQGMO_SET_SIGNAL
MQGMO_SYNCPOINT
MQGMO_MARK_SKIP_BACKOUT
MQGMO_MSG_UNDER_CURSOR
MQGMO_LOCK
MQGMO_UNLOCK
MQGMO_LOGICAL_ORDER
MQGMO_COMPLETE_MSG
MQGMO_ALL_MSGS_AVAILABLE
MQGMO_ALL_SEGMENTS_AVAILABLE.

5.4.3 MQCLOSE options to process unread messages

At the time a program requests closure of a queue which has read ahead enabled there may be non-persistent messages which have been destructively removed from the queue manager but are still held in memory on the client system.

Two new options on the MQCLOSE verb allow the program to determine what is done with these messages:

MQCO_IMMEDIATE specifies that the messages are to be discarded. This is the default behavior if no options are specified.

MQCO_QUIESCE specifies that if there are any messages in memory the MQCLOSE returns a Completion Code of MQCC_WARNING and a Reason Code of MQRC_READ_AHEAD_MSGS. The program can test for these and issue calls to get the messages until there are none left in memory. MQ does not attempt to read ahead any more messages from the queue manager. The object handle to the open queue remains valid until MQCLOSE has been called again to close the queue.

5.4.4 Displaying connection status of read ahead

The MQ Explorer and MQSC command DISPLAY CONN have been enhanced to show the status of read ahead on connected programs. The READA field may display the following values:

YES indicates that read ahead is enabled on a open queue and is being used efficiently.

NO indicates that read ahead is not enabled on any open queues.

INHIBITED indicates that read ahead was requested by the program but has been inhibited because of incompatible options specified on the first call to MQGET.

BACKLOG indicates that read ahead is enabled but is not being used efficiently. Non-persistent messages have been streamed into memory on the client system but the client program is not requesting them. For example, the program may have started using MQGET for specific Correlation Ids.

Refer to Chapter 8, “Administration enhancements” on page 151 for an example of MQ Explorer being used to display connection status.

5.5 Asynchronous put

This new feature of WebSphere MQ V7.0 is available in all programming environments but it is of most benefit to MQ Client and Java / JMS programs which require interaction with the queue manager over a communications link. It allows messages to be put to a queue without waiting for a status response from the queue manager containing the Completion Code and Reason Code. The status response can be obtained after the messages have all been put.

When syncpointing is not being used this provides a lower Quality of Service (QoS) because the program cannot be sure that all messages were put successfully if the connection fails before the status response has been checked. If the connection fails during a syncpoint Unit Of Work (UOW) the messages are backed out and nothing is put to the queue.

Non-persistent and persistent messages are supported. Persistent messages must be put with syncpoint to take advantage of asynchronous put, as persistent messages which are put without syncpoint require a confirmation response from the queue manager before the program can continue.

This elimination of the synchronous interaction between client and queue manager for each message provides a significant improvement in throughput in the right circumstances. Figures 5-4 and 5-5 show the interactions required without asynchronous put and then with it.

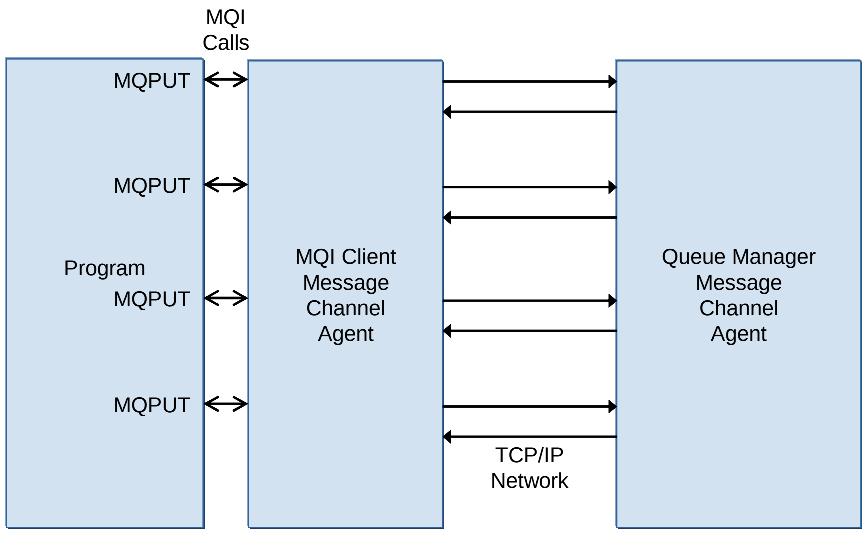

MQI Calls
MQPUT
MQPUT
Program
MQPUT
MQPUT
MQI Client Message Channel Agent
Queue Manager Message Channel Agent
TCP/IP Network

Figure 5-4 Client putting messages without asynchronous put

An example of using this feature is a client program which needs to load a large number of transaction messages on to a queue and then display a confirmation to the program user as quickly as possible.

The program issues a sequence of calls to MQPUT or MQPUT1. After it has put the messages, it issues a call to MQSTAT to get the status information. If a unit of work was in progress and persistent messages were put with syncpoint, a call to MQCMIT made prior to MQSTAT succeeds if all of the asynchronous put operations succeeded.

After a message has been put asynchronously some of the output fields in the Message Descriptor (MQMD), such as Message Id, and output fields in the put options, are not updated. This because they are part of the response from the queue manager which has not yet been received.

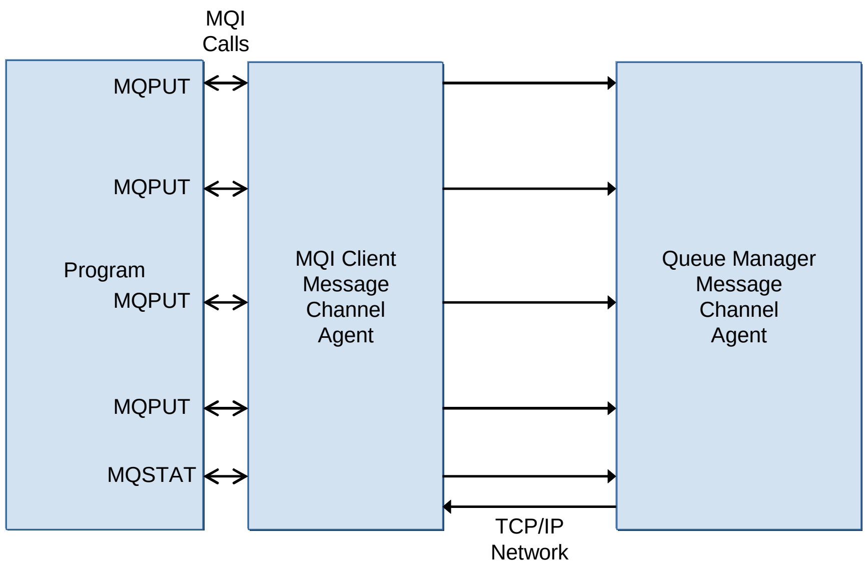


Figure 5-5 Client putting messages with asynchronous put

Asychronous put does not affect the relationship or sequencing between putting and getting messages when compared with previous versions of WebSphere MQ. For example, getting a message with a specific Correlation Id is delayed until all prior asynchronous puts on that queue have been completed.

5.5.1 MQPUT and MQPUT1 options for asynchronous put

There are four three options on the MQPUT and MQPUT1 verbs which control the asynchronous put feature on normal queue objects:

MQPMO_RESPONSE_AS_QDEF is the default if none of the options is specified.

On the MQPUT verb MQPMO_RESPONSE_AS_QDEF specifies that the behavior for putting the message is determined by the setting of the new parameter DEFPRESP on the queue at the time it was opened.

The DEFPRESP parameter may have the following values:

SYNC indicates that the message is to be put synchronously. See the description of MQPMO_SYNC_RESPONSE below for details.

ASYNC indicates that the message is to be put asynchronously. See the description of MQPMO_ASYNC_RESPONSE below for details.

On the MQPUT1 verb the syncpoint options select whether asynchronous put or synchronous put is used. MQPMO_SYNCPOINT implies MQPMO_ASYNC_RESPONSE, and MQPMO_NO_SYNCPOINT implies MQPMO_SYNC_RESPONSE.

MQPMO_SYNC_RESPONSE specifies that the message is to be put synchronously. The call returns the correct Completion Code and Reason Code response. This is equivalent to the behavior in versions prior to WebSphere MQ V7.0.

MQPMO_ASYNC_RESPONSE specifies that message is to be put asynchronously. The call returns a successful Completion Code and Reason Code response, unless there is an error which is detected prior to passing the message to the queue manager. The program later needs to call MQSTAT to obtain the actual status response.

There is a fourth option which only applies to a topic object which has been opened to put published messages. MQPMO_RESPONSE_AS_TOPIC_DEF is equivalent to MQPMO_RESPONSE_AS_Q_DEF.

Only one of these parameters can be specified as an option to MQPUT or MQPUT1.

If the client program is connected to a queue manager which is prior to version 7.0, the message is always put synchronously, regardless of the option settings.

5.5.2 MQSTAT to obtain status of asynchronous puts

The new MQSTAT verb returns information about the status of all asynchronous puts which have been made since the connection was established or the most recent call to MQSTAT. The format of the call and details of the new MQSTS data type are fully described in Chapter 6, "Message Queue Interface extensions" on page 87.

The following example contains code in the C language which puts some message asynchronously and then uses MQSTAT to retrieve the status information:

Example 5-2 C code to retrieve status information using MQSTAT after calls to MQPUT

```
int MsgIdx, NumMsgs;
MQBYTE *MyMsgDataPtr[MYMAXMSGS];
MQLONG MyMsgDataLen[MYMAXMSGS];
HCONN hConn;
HOBJ hObj;
MQMD MsgDesc;
```

```
MQLONG PutMsgOpts;
MQLONG CompCode, ReasCode, StatType = MQSTAT_TYPE_ASYNC_ERROR;
MQSTS StatInfo = MQSTS_DEFAULT;
...
   printf( “Putting %d messages.\n”, NumMsgs );
   for( MsgIdx = 0 ; MsgIdx < NumMsgs ; MsgIdx++ )
   {
      MQPUT( hConn, hObj, &MsgDesc, &PutMsgOpts, MyMsgDataLen[MsgIdx],
         MyMsgDataPtr[MsgIdx], &CompCode, &ReasCode );
      if( CompCode != MQCC_OK )
      {
      printf( “Failed to put message %d, Completion Code %d, ”
         “Reason Code %d.\n”, CompCode, ReasCode );
      ... (take appropriate action)
      }
   }
   MQSTAT( hConn, StatType, &StatInfo, &CompCode, &ReasCode );
   if( CompCode == MQCC_OK )
      printf( “All completed OK\n” );
   else
      {
      printf( “Failed with Completion Code %d, Reason Code %d.\n”,
         CompCode, ReasCode );
      printf( “%d puts succeeded, %d had warnings, %d failed.\n”,
         StatInfo.PutSuccessCount, StatInfo.PutWarningCount,
         StatInfo.PutFailureCount );
      printf( “The first error was Completion Code %d, ”
         “Reason Code %d.\n”, StatInfo.CompCode, StatInfo.Reason );
      ... (take appropriate action)
      }
   ...
```

5.6 Instance limits on SVRCONN channels

The SVRCONN type channel has been enhanced in WebSphere MQ V7.0 to add two new parameters which limit the number of concurrently running instances of each channel.

The main reason for providing this limit is to prevent client programs from running the maximum number of channel instances available on a queue manager, or to prevent programs from using more running channel instances that would reasonably be expected in normal operation. An excessive number of running channels is usually due to a design flaw, a programming bug, or a deliberate attempt to deny service on the queue manager.

Each queue manager has a parameter which set the maximum number of running instances across all types of channels. During the period that this maximum number of channels is running on the queue manager it is not possible to start new instances of any type of channel. This could have a severe impact on business functions which use the queue manager. Therefore it is very useful to able to limit the distribution of maximum running channels on a individual SVRCONN channels, leaving capacity for other types of channels to start when they need to.

Refer to the manual *WebSphere MQ System Administration,* SCXX-XXXX-XX for further details on the MaxChannels, MaxActiveChannels, MAXCHL and MAXTCP parameters which set the absolute maximums for the queue manager. If the maximum was not specified when the queue manager was created, or changed afterwards, it defaults to 100 on distributed platforms and 200 on the z/OS platform. This maximum needs to be increased if there are a larger total number of client channels and other types of channel which need to run simultaneously.

The two new parameters on the SVRCONN type channel are described in the following sections.

5.6.1 MAXINST

The value of this parameter is the maximum number of instances which can be run by all client programs connecting to the queue manager via the channel. A client program which use multiple threads with the new conversation sharing feature does not increase the number of running channels, as its threads all share a single running channel instance.

A value of 0 indicates that none can run.

A value of 99999999 indicates that an unlimited number can run. This is the default for new channels or channels which have been migrated from a previous version to WebSphere MQ V7.0. The default for new channels may have changed if the MAXINST parameter on the SYSTEM.DEF.SVRCONN channel has been altered since the queue manager was created.

Careful consideration should be made before setting this parameter to an intermediate value. The normal number of running channel instances should be observed during peak periods and a future projection made using capacity planning information. A first approximation is to double the number of running instances observed in peak periods. This should be observed on a regular basis so that growth is foreseen and the parameter value adjusted before it is reached unintentionally during normal operations.

5.6.2 MAXINSTC

The value of this parameter is the maximum number of instances which can be run by all client programs connecting to the queue manager via the channel from a unique network address. A client program which use multiple threads with the new conversation sharing feature does not increase the number of running channels, as its threads all share a single running channel instance.

A value of 0 indicates that none can run.

A value of 99999999 indicates that an unlimited number can run from a unique network address. This is the default for new channels or channels which have been migrated from a previous version to WebSphere MQ V7.0. The default for new channels may have changed if the MAXINSTC parameter on the SYSTEM.DEF.SVRCONN channel has been altered since the queue manager was created.

The same planning considerations should be made as per the MAXINST parameter before setting the MAXINSTC parameter to an intermediate value. It should be set above the peak number of connections made by a unique client system, which could be quite a low value.

5.6.3 Dynamic changes

If a channel is altered to reduce the value of the MAXINST or MAXINSTC parameters to less than the current number of running instances of the channel, it does not affect the client programs which are currently connected. However, new instances are not able to run until the total number of running instances has reduced below the new values.

5.6.4 Examples of setting the new parameters

The following MQSC command would be used to set the instance limits on the APP1.CLIENT channel, where there are known to be 5 systems on which client programs run and use this channel, and up to 20 instances of the program would normally run on each system:

```
ALTER CHANNEL(APP1.CLIENT) CHLTYPE(SVRCONN) MAXINST(100) MAXINSTC(20)
```

The following MQSC command would be used to set the instance limits on the APP2.CLIENT channel, where there are known to be 2,000 systems on which a client program can run and use this channel, and only one instance of the program is allowed to run on each system:

```
ALTER CHANNEL(APP2.CLIENT) CHLTYPE(SVRCONN) MAXINST(2000) MAXINSTC(1)
```

Refer to Chapter 8, “Administration enhancements” on page 151 for an example of MQ Explorer being used to set these parameters on a SVRCONN channel.

5.7 Weighted selection on CLNTCONN channels

In recent versions of WebSphere MQ a client program can request connection to a queue manager name which is prefixed by an asterisk character:

```
MQCONN( “*SALES”, &hConn, &CompCode, &ReasCode );
```

This feature is demonstrated in the section “Examples of using MQCONN calls” of the manual *WebSphere MQ Clients,* SCXX-XXXX-XX.

The asterisk instructs MQ to attempt connection using a set of channel names which are defined in a Client Channel Definition Table (CCDT). The algorithm builds an alphabetic list of channel names which match the queue manager name. It then attempts to connect using the first channel in the list. If this does not succeed it tries the next one, and so on.

The typical application of this simple algorithm is to provide a secondary “back up” queue manager which is used when the primary queue manager is not available.

New parameters have been added to the CLNTCONN type channel in WebSphere MQ V7.0 to allow more intelligent selection based on a relative weighting and availability of queue managers. Simple selection by alphabetic precedence and random selection based on a numeric weighting factor are both provided. This extends the algorithm to cater for work-load balancing across multiple queue managers.

5.7.1 CLNTWGHT parameter

The first new parameter on the CLNTCONN type channel is CLNTWGHT (Client Weight). A value of 0 indicates the channel has no weighting. A value in the range 1 to 99 indicates a weighting which is relative to other channels which have a non-zero weighting and also match the queue manager name.

The new algorithm first considers the matching channels which have no weighting. Using the same logic as the old algorithm, MQ attempts connection via the channels in alphabetic order. It then considers the other matching channels which have a non-zero weighting. MQ initially populates a list by randomly selecting all of these channels. The probability of selecting a channel is its CLNTWGHT value divided by the sum of this value across all matching channels.

If a connection attempt fails via the first channel on the list, it is moved to the end of the list, and the next one on the list is tried, and so on.

The following example contains MQSC commands to define three client channels. It demonstrates an equal probability of connection among three queue managers if the queue manager name is specified as "*SALES" in the connection request:

```
DEFINE CHANNEL(QMA.CLIENT) CHLTYPE(CLNTCONN) TRPTYPE(TCP)
CONNAME('sysa.rb.com') QMNAME(SALES) CLNTWGHT(50)

DEFINE CHANNEL(QMB.CLIENT) CHLTYPE(CLNTCONN) TRPTYPE(TCP)
CONNAME('sysb.rb.com') QMNAME(SALES) CLNTWGHT(50)

DEFINE CHANNEL(QMC.CLIENT) CHLTYPE(CLNTCONN) TRPTYPE(TCP)
CONNAME('sysc.rb.com') QMNAME(SALES) CLNTWGHT(50)
```

The real names of the queue managers on these systems do not need to be "SALES". It is merely used to identify a set of queue managers in a CCDT. The name is not checked against the actual queue manager when the connection is being established.

5.7.2 AFFINITY

The second new parameter on the CLNTCONN channel is AFFINITY and its value may be PREFERRED or NONE. This parameter determines the channel selection criteria which is used for multiple connection requests made by a single process. The default value is PREFERRED, although this may be changed by altering the SYSTEM.DEF.CLNTCONN channel after MQ has been installed.

PREFERRED uses the new algorithm for the first connection request, as explained above. Subsequent connection requests by the process reuse the list of channels generated by the first request. This means that the same "preferred" channel and queue manager is used, provided the connection can be successfully made. If a connection attempt fails the list is re-ordered as per the algorithm and the updated list is reused by subsequent connection requests.

NONE uses the new algorithm and regenerates the list of channels for each connection request. This means that weighted channels are selected randomly each time and are not related to the channels which are used by other active connections in the process.

5.8 Reconnecting via a previously used channel

A client program can make connection request to a queue manager name that is blank, prefixed by an asterisk character (*), or only an asterisk. MQ may then establish the connection by one of several possible channels in the Client Channel Definition Table (CCDT). The actual channel name and connected queue manager name is not under direct control of the client program.

In a particular application design, it may be necessary for a client program to reconnect to the same queue manager via the same channel which was used previously.

A new feature in WebSphere MQ V7.0 allows a successful call to MQCONNX to return the channel name in a channel definition data type.

The channel name can then be saved and later set in a channel definition data type and passed to a subsequent call to MQCONNX. This enforces a connection via a specific channel and hence to a specific queue manager. It bypasses the normal selection algorithm which is used to process CCDT files.

The following example contains code in the C language which saves the channel name from the channel definition after a successful call to MQCONNX:

Example 5-3 C code to save the channel name after successful MQCONNX

```
#include <cmqc.h>
#include <cmqxc.h> /* For MQCD structure */
...
MQHCONN hConn = MQHC_UNUSABLE_HCONN;
MQCNO ConnOpts = MQCNO_DEFAULT;
MQLONG CompCode, ReasCode;
MQCD ChlDef = MQCD_DEFAULT;
MQCHAR MyChlName[MQ_CHANNEL_NAME_LENGTH];
...
   ConnOpts.Version = MQCNO_VERSION_2; /* The default version 1 does
    not support ClientConnOffset and ClientConnPtr fields */
   ConnOpts.Options = MQCNO_CD_FOR_OUTPUT_ONLY;
   ConnOpts.ClientConnOffset = 0;
   ConnOpts.ClientConnPtr = &ChlDef;
   MQCONNX( “*SALES”, ConnOpts, &hConn, &CompCode, &ReasCode );
   if( CompCode == MQCC_OK )
```

```
    { /* ChlDef contains a valid ChannelName, save it */
      memcpy( MyChlName, ChlDef.ChannelName, MQ_CHANNEL_NAME_LENGTH );
      ...
```

A program can set the channel name and use it to establish a connection to a predictable queue manager. The following code demonstrates this as a continuation of the code in Example 5-2:

Example 5-4 C code to use a specific channel name for MQCONNX

```
...
   ConnOpts.Version = MQCNO_VERSION_2; /* The default version 1 does
    not support ClientConnOffset and ClientConnPtr fields */
   ConnOpts.Options = MQCNO_USE_CD_SELECTION;
   ConnOpts.ClientConnOffset = 0;
   ConnOpts.ClientConnPtr = &ChlDef;
   memcpy( ChlDef.ChannelName, MyChlName, MQ_CHANNEL_NAME_LENGTH );
   MQCONNX( "*SALES", ConnOpts, &hConn, &CompCode, &ReasCode );
   if( CompCode == MQCC_OK )
      ...
```

If the MQSERVER environment variable has been set and it matches the channel name, the channel configuration details from the value of this variable is used (transport type, host name, port number). Otherwise, the configuration details from the CCDT is used.

5.9 Maximum message length increased on MQSERVER environment variable

The MQSERVER environment variable can be used to define a minimal CLNTCONN type channel to allow a MQ Client program to connect to a queue manager. This is an alternative to using a Client Channel Definition Table (CCDT) or a programmatic method such as MQCONNX in the MQI to define the channel.

Only the channel name, transport protocol and the connection name (host name and port number) can be specified in the value of this variable, so the configuration parameters are very limited.

In WebSphere MQ V7.0 a channel defined using MQSERVER has a maximum message length (MAXMSGL) of 100MB (104,857,600 bytes). This has been increased from the 4MB (4,194,304 bytes) maximum message length in previous versions of MQ.

To take advantage of the increased maximum length the identically named SVRCONN type channel must also have its MAXMSGL altered to an appropriate value on the queue manager. The default value taken for MAXMSGL when a SVRCONN channel is defined is 4MB, although this may have changed on the SYSTEM.DEF.SVRCONN channel object after MQ was installed.

5.10 Security exit details in WebSphere MQ Explorer

Security Exits can now be directly enabled and configured in MQ Explorer for each of the client channel connections to remote queue managers.

The exit code must be in a Dynamic Link Library (DLL) and written in C, or in a Java Archive (JAR) file and written in Java. Data for the security exit can also be configured.

Previously in WebSphere MQ V6.0 a security exit and its data could only be specified for use with MQ Explorer by defining the CLNTCONN channel in a Client Channel Definition Table (CCDT) file. This method continues to be supported in V7.0.

The procedure for configuring Security Exits and other security related features is described in Chapter 8, “Administration enhancements” on page 151.

5.11 Using MQ Explorer without a Client Attach Facility (CAF) license on z/OS

A limited capacity is provided for MQ Explorer and other MQ administration programs to administer z/OS queue managers without the need to purchase a CAF license for WebSphere MQ V7.0 on z/OS. Up to 5 instances of these programs can connect to each z/OS queue manager at version 7.0 via the Client channel “SYSTEM.ADMIN.SVRCONN” without this license.

Refer to Chapter 9, “Publish/Subscribe management” on page 193 for further details on enabling this capability.

5.12 Compatibility

Version 6.0 and 7.0 WebSphere MQ Clients can connect to version 6.0 and 7.0 WebSphere MQ queue managers. The base features of WebSphere MQ Client version 6.0 are available in all four combinations, but most of the new features of version 7.0 are only available when a version 7.0 client connects to a version 7.0 queue manager.

The new features which are available for a version 7.0 client connected to a version 6.0 queue manager are:

- Weighted selection on CLNTCONN channels.
- Reconnecting via a previously used channel.
- Maximum message length increased on MQSERVER environment variable.

The new features which are available for a version 6.0 client connected to a version 7.0 queue manager are:

- Instance limits on SVRCONN channels.

CLNTCONN channel definitions in a Client Channel Definition Table (CCDT) file which have been created or altered using WebSphere MQ V7.0 cannot be used with previous versions of WebSphere MQ. This is because the channel definition information in the file has been extended due to enhancements in version 7.0 and this is not understood by previous versions.

6

Message Queue Interface extensions

This chapter describes the Message Queue Interface (MQI) extensions that are introduced in WebSphere MQ V7.0.

The description of the function calls and data structures in this chapter is at a level that facilitates the understanding of the new capabilities. For detailed descriptions of the functions, parameters and data structures, refer to *Application Programming Reference,* SCXX-XXXX-XX manual.

The chapter covers the following new functionality in WebSphere MQ V7.0 and includes the following topics:

- “Variable Length Strings”
- “Message properties”
- “Message browsing”
- “Call Back for asynchronous consumers”
- “Publish/Subscribe”
- “Put action indicators”
- “Message selectors”
- “Other MQI considerations”

6.1 Variable Length Strings

WebSphere MQ V7.0 data structures uses variable length strings therefore the reader should understand how they are implemented before reading about other more exciting features of the Message Queue Interface.

Some data structures in WebSphere MQ V7.0 include a new data type called MQCHARV to represent variable length strings such as topic strings, object names, subscriber user data, selection strings and other new data elements.

MQCHARV is a data structure that describes the location and attributes of a variable string. MQCHARV is used to pass or to receive variable strings as parameters to or from WebSphere MQ V7.0 queue manager.

There are two methods to specify the location of the buffer containing a variable length string: using a pointer or using an offset. The offset method is intended for programming languages like COBOL that in some platforms do not support the use of pointers.

There are also two methods to specify the length of the string: to use the actual string length value or to specify that the string is null terminated.

Applications need to allocate a buffer to receive or to send variable length strings from or to WebSphere MQ V7.0 queue manager. If a buffer is not specified to receive a variable length string, it is not returned to the application after the execution of Message Queue Interface function call.

6.1.1 MQCHARV data structure

The following table shows the fields of the MQCHARV data structure:

Table 6-1 MQCHARV data structure

Field name	Type	Default value	Description
VSPtr	MQPTR	Null	Pointer to string buffer or null when offset is used
VSOffset	MQLONG	0	Offset to string buffer or zero
VSBufSize	MQLONG	MQVS_USE_VSLENGTH	String buffer size
VSLength	MQLONG	0	String length or MQVS_NULL_TERMINATED
VSCCSID	MQLONG	MQCCSI_APPL	CCSID used to encode the string

6.1.2 MQCHARV using pointer

The following are the steps to define a MQCHARV data structure with a pointer to the variable string:

1. Allocate a buffer for the variable string and set the buffer size in VSBufSize.
2. Set in VSPtr the pointer to the string buffer.
3. If a string is provided as input to a Message Queue Interface function call then the string is stored in the buffer and the length is stored in VSLength. If the Message Queue Interface function call is returning a string to the application then VSLength should have the value of zeros and the string buffer must be empty.
4. If the string is passed as input to function call the CCSID that corresponds to the character set encoding of the string must be set in VSCCSID. If the string is returned to the application program the required CCSID must be set on VSCCSID as input. The queue manager tries to convert the string to the required CCSID but if it is not possible it returns in VSCCSID the CCSID of the string.

The following figure shows MQCHARV using a pointer:

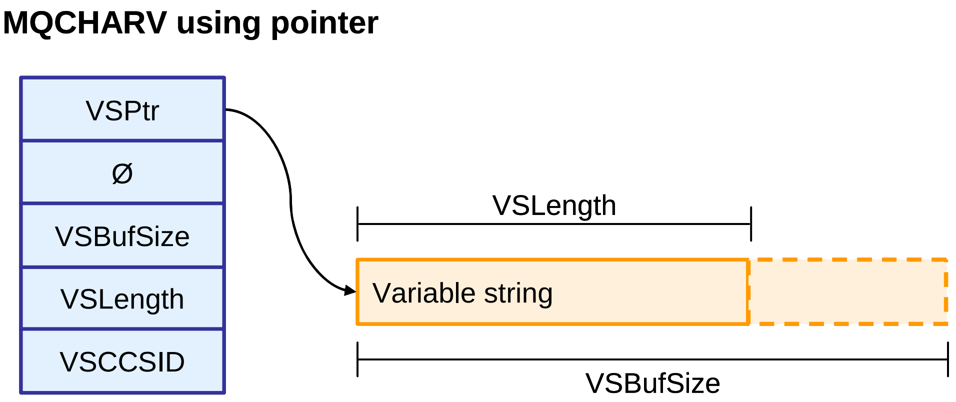


Figure 6-1 MQCHARV using pointer

6.1.3 MQCHARV with offset

The following are the steps to define a MQCHARV data structure with an offset to the string buffer:

1. Allocate a buffer for the variable string and set the buffer size in VSBufSize.
2. Set VSPtr to null to indicate that offset is used instead of a pointer.

3. Calculate the offset from the beginning of MQCHARV to the beginning of the string buffer. The offset can a positive or negative value.
4. If a string is provided as input to the queue manager then store the string in the buffer and set the length in VSLength. If a string is returned to the application then leave VSLength with value zero and leave the string buffer empty.
5. If the string is passed as input to the queue manager set the CCSID that corresponds to the character set of the string. If the string is returned to the application program set the required CCSID on VSCCSID. The queue manager tries to convert the string to the required CCSID but if it is not possible it returns in VSCCSID the CCSID of the string.

The following figure shows a MQCHARV data structure with offset:

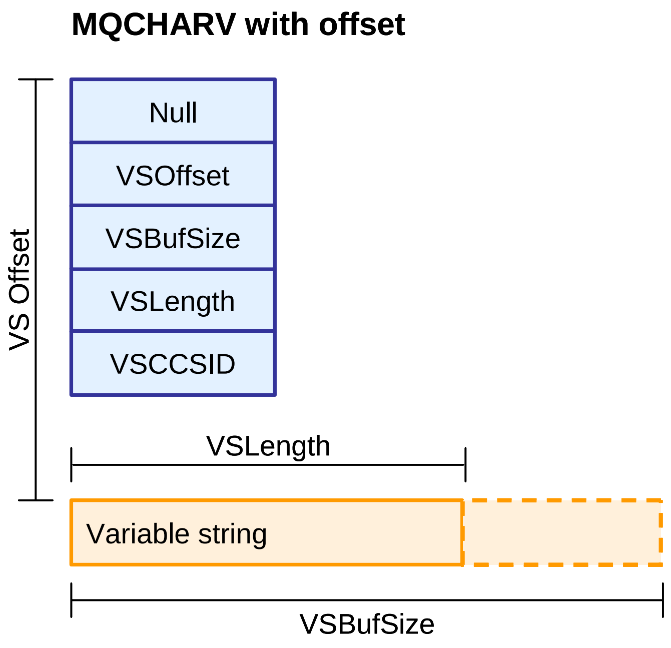


Figure 6-2 MQCHARV with offset

6.1.4 Null terminated strings

If the variable length string is null terminated then the value MQVS_NULL_TERMINATED may be specified in VSLength.

The following figure shows a MQCHARV with a null terminated string:

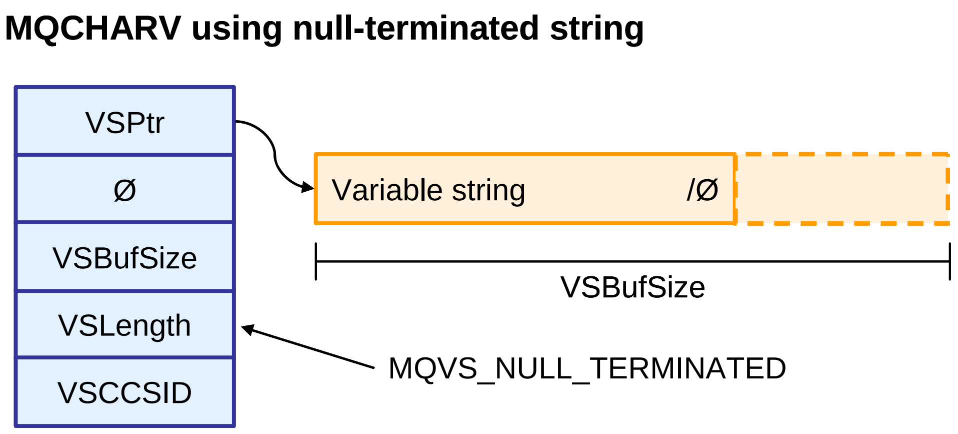


Figure 6-3 MQCHARV using null terminated string

6.1.5 Coded Character set identifier

Each variable length string requires the specification of the coded character set identifier (CCSID) that encoded the string. This enables applications to send and receive variable length strings on expected character sets. The CCSID is specified in VSCCSID. The default value MQCCSI_APPL indicates that the strings are encoded in the default CCSID of the Queue manager for server binding connections or of the MQ Client system for client binding connections.

If the application is running with a different character set from the Queue manager or from the MQ Client then the application needs to override the value of the constant MQCCSI_APPL with the CCSID required. This initializes all the MQCHARV structures with the correct CCSID value.

6.2 Message properties

Message properties are used by message selectors to filter publications to topics or to selectively get messages from queues. Message properties can be used to include business data or state information without having to store it in the message data.

Another benefit of message properties is that applications do not have to access data in the MQ Message Descriptor (MQMD) or RFH headers because fields in these data structures can be accessed as message properties using the Message Queue Interface functions calls described in this section.

6.2.1 Message handles

Message handles are new in WebSphere MQ V7.0 and they are references to the message properties of a message.

Message properties are also new in WebSphere MQ V7.0 and they are name-value pairs that are part of messages. Java Message Service (JMS) introduced the concept of message properties and now they are supported in WebSphere MQ V7.0 as part of any message.

There are new Message Queue Interface (MQI) functions to create and delete message handles (MQCRTMH and MQDTLMH). The message handle is used on the MQPUT and MQPUT1 calls to associate message properties to the message being put. Similarly a message handle in the MQGET function is the reference to all the message properties when the MQGET is complete.

The scope of message handles is between the MQCRTMH function call and the MQDLTMH function call or the unit of processing that defines the handle.

6.2.2 Set, inquire and delete message properties

There are new Message Queue Interface function calls are used to set, inquire and delete message properties (MQCRTMP, MQINQMP and MQDLTMP). These function calls require a message handle as one of the parameters.

6.2.3 MQCRTMH create message handle

This function call creates a new message handle that can used to create or to inquire message properties.

Syntax

This is the MQCRTMH function syntax:

```
MQCRTMH (Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason)
```

Parameters

These are the parameters used by MQCRTMH:

Table 6-2 MQCRTMH parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Connection to the Queue manager

Parameter name	Data type	Input/Output	Description
CrtMsgHOpts	MQCMHO	Input	Options to control how message handles are created
Hmsg	MQHMSG	Output	Message handle
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

This example shows how to create a message handle for a unassociated connection and another message handle associated with a connection.

Example 6-1 MQCRTMH C language example

```
/* Define data structures and initialize with default values. */

MQHCONN hConn       = MQHC_UNUSABLE_HCONN;
MQLONG  CompCode    = MQCC_OK;
MQLONG  Reason      = MQRC_NONE;
MQCMHO  CrtMsgHOpts = {MQCMHO_DEFAULT};
MQHMSG  hMsg        = MQHM_UNUSABLE_HMSG;

/* Connect to the queue manager */
 MQCONN(“QM”, &hConn, &CompCode, &Reason);

/* Case 1: Create a message handle unassociated with a connection. */

MQCRTMH(MQHC_UNASSOCIATED_HCONN, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);

/* Case 2: Create a new message handle associated with a specific connection. */

MQCRTMH(hConn, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);
```

Note: Message handles created with MQHC_UNASSOCIATED_HCONN can be used on MQGET/MQPUT function calls for any connection within the unit of processing, but the message handle may be used by only one such MQI call at a time.

Message handles associated with a specific Hconn must be used with MQI calls for that specific connection.

6.2.4 MQDLTMH delete message handle

This function call deletes a message handle.

Syntax

This is the MQDLTMH function syntax:

```
MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason)
```

Parameters

These are the MQDLTMH parameters:

Table 6-3 MQDLTMH parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Represents a valid Queue manager connection. It must be the same connection used when handle was created. If the handle was un-associated then this must be a valid connection to be able to delete the handle.
Hmsg	MQHMSG	Input/Output	Handle to be deleted and on successful completion the handle is set to MQHM_UNUSABLE_HMSG.
DltMsgHOpts	MQDMHO	Input	Options to control how the handle is deleted.
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

This example shows how to create and delete a message handle.

Example 6-2 MQDLTMH C language example

```
/* Define data structures and initialize with default values. */

MQHCONN hConn       = MQHC_UNUSABLE_HCONN;
MQLONG  CompCode    = MQCC_OK;
MQLONG  Reason      = MQRC_NONE;
MQCMHO  CrtMsgHOpts = {MQCMHO_DEFAULT};
MQHMSG  hMsg        = MQHM_UNUSABLE_HMSG;
MQDMHO  DltMsgHOpts = {MQDMHO_DEFAULT};

/* Connect to the queue manager */
```

```
MQCONN("QM", &hConn, &CompCode, &Reason);

/* Create a new message handle */

MQCRTMH(hConn, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);

/* Delete the message handle just created */

MQDLTMH(hConn, &hMsg, &DltMsgHOpts, &CompCode, &Reason);
```

6.2.5 MQSETMP set a message property

This function call sets or modifies a message property that is referenced by a message handle.

Syntax

This is the MQSETMP function syntax:

```
MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,
Value, CompCode, Reason)
```

Parameters

These are the MQSETMP parameters:

Table 6-4 MQSETMP parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the Queue manager.
Hmsg	MQHMSG	Input	Message handle to which the new message property is added.
SetPropOpts	MQSMPO	Input	Options to control how the message property is added.
Name	MQCHARV	Input	Name of the property. Names must follow rules described in the product documentation.
PropDesc	MQPD	Input/Output	To define attributes of a property. This parameter is intended for advanced uses of message properties.

Parameter name	Data type	Input/Output	Description
Type	MQLONG	Input	This is the data type of the property. Such as boolean, integer, float, string and null.
ValueLength	MQLONG	Input	Length in bytes of the property value. The constant MQVL_NULL_TERMINATED can be used for null terminated strings.
Value	MQBYTEx	Input	Buffer containing the value of the property
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

This example shows how to set message properties of various types.

Example 6-3 MQSETMP C language example

```
/* Define data structures and initialize with default values. */

MQHCONN hConn       = MQHC_UNUSABLE_HCONN;
MQLONG  CompCode    = MQCC_OK;
MQLONG  Reason      = MQRC_NONE;
MQCMHO  CrtMsgHOpts = {MQCMHO_DEFAULT};
MQHMSG  hMsg        = MQHM_UNUSABLE_HMSG;
MQPD    PropDesc    = {MQPD_DEFAULT};
MQSMPO  SetPropOpts = {MQSMPO_DEFAULT};
MQCHARV Name        = {MQCHARV_DEFAULT};
MQINT32 IntValue    = 5;

/* Connect to the queue manager */

MQCONN(“QM”, &hConn, &CompCode, &Reason);

/* Create a new message handle */

MQCRTMH(hConn, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);

Name.VSPtr = “Sport”;
Name.VSLength = MQVS_NULL_TERMINATED;

/* Set a null-terminated string property value */

MQSETMP( hConn, hMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_STRING,
        MQVL_NULL_TERMINATED, “football”, &CompCode, &Reason );
```

```
Name.VSPtr = “Goals”;

/* Set an integer property value */

MQSETMP( hConn, hMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_INT32,
        4, &IntValue, &CompCode, &Reason );

Name.VSPtr = “Money”;

/* Set a null property value */

MQSETMP( hConn, hMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_NULL,
        0, NULL, &CompCode, &Reason );


/* Set an empty property value */

MQSETMP( hConn, hMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_STRING,
        0, NULL, &CompCode, &Reason );

/*****************************************************/
/* Note that after the four calls to MQSETMP         */
/* the message handle has only three properties in it:*/
/* “Sport” with the value “football”,                */
/* “Goals” with the value “5”, and                   */
/* “Money” with the value “”.                        */
/*****************************************************/
```

6.2.6 MQINQMP inquire message property

This function call returns the value of a property referenced by a message handle.

The wildcard character “%” may be used at the end of property names to inquire for multiple properties. The MQINQMP returns one value at a time therefore Inquire Property Options (InqPropOpts) are used to inquire for the first (MQIMPO_INQ_FIRST) and subsequent property values (MQIMPO_INQ_NEXT). Multiple MQINQMP calls are required to inquire all the properties that match a wildcard. Reason code MQRC_PROPERTY_NOT_AVAILABLE is returned when no more matching properties are available.

If ValueLength is less than the length of the property value a reason code MQRC_PROPERTY_VALUE_TOO_BIG is returned. The actual length is returned in the DataLength parameter and the MQINQMP call can be reissued with the correct ValueLength. It is also possible to inquire the property length using MQIMPO_QUERY_LENGTH option this returns the length without error.

The CCSID and data encoding of the property value is returned in the InqPropOpts parameter in the fields ReturnedCCSID and ReturnedEncoding.

Syntax

This is MQINQMP function syntax:

```
MQINQMP(Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type, ValueLength,
Value, DataLength, CompCode, Reason)
```

Parameters

These are the MQINQMP parameters:

Table 6-5 MQINQMP parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the Queue manager.
Hmsg	MQHMSG	Input	Message handle to which the new message property is added.
InqPropOpts	MQIMPO	Input	Options to control how the message properties are inquired.
Name	MQCHARV	Input	Name of the property to inquire. The property name may end with the wildcard character “%” to inquire multiple properties.
PropDesc	MQPD	Output	To define attributes of a property. This parameter is intended for “advanced” uses of message properties. Most applications are not interested in setting these options.
Type	MQLONG	Input/Output	This is the data type of the property. For example boolean, integer, float, string and null.
ValueLength	MQLONG	Input	Length in bytes of the Value area. If zero is specified then no value is returned.
Value	MQBYTEx	Output	Area to contain the returned value of the property.

Parameter name	Data type	Input/Output	Description
DataLength	MQLONG	Output	The actual length in bytes of the property value. If ValueLength is smaller than DataLength then MQINQMP can be reissued with the correct ValueLength.
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

This example shows how to create a message handle, set a message property and inquire the message property.

Example 6-4 MQINQMP C language example

```
/* Define data structures and initialize with default values. */

MQHCONN hConn       = MQHC_UNUSABLE_HCONN;
MQLONG  CompCode    = MQCC_OK;
MQLONG  Reason      = MQRC_NONE;
MQCMHO  CrtMsgHOpts = {MQCMHO_DEFAULT};
MQHMSG  hMsg        = MQHM_UNUSABLE_HMSG;
MQPD    PropDesc    = {MQPD_DEFAULT};
MQSMPO  SetPropOpts = {MQSMPO_DEFAULT};
MQCHARV Name        = {MQCHARV_DEFAULT};
MQIMPO  InqPropOpts = {MQIMPO_DEFAULT};
MQBYTE  Value[256];
MQLONG  Type        = MQTYPE_AS_SET;
MQLONG  DataLength;

/* Connect to the queue manager */

MQCONN(“QM”, &hConn, &CompCode, &Reason);

/* Create a new message handle */

MQCRTMH(hConn, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);

Name.VSPtr = “Sport”;
Name.VSLength = MQVS_NULL_TERMINATED;

/* Set a null-terminated property value */

MQSETMP( hConn, hMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_STRING,
        MQVL_NULL_TERMINATED, “Football”, &CompCode, &Reason );

/* Inquire the property value */

MQINQMP( hConn, hMsg, &InqPropOpts, &Name, &PropDesc, &Type, sizeof(Value),
        Value, &DataLength, &CompCode, &Reason );
```

6.2.7 MQDLTMP delete message property

This function call deletes a message property from a message handle.

Syntax

This is the MQDLTMP function syntax:

```
MQDLTMP (Hconn, Hmsg, DltPropOpts, Names, CompCode, Reason)
```

Parameters

These are the MQDLTMP parameters:

Table 6-6 MQDLTMP parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the Queue manager.
Hmsg	MQHMSG	Input	Message handle from which message property is deleted.
DltPropOpts	MQDMPO	Input	Options to control how the message property is deleted.
Name	MQCHARV	Input	Name of the property to delete. Wildcard characters are not allowed.
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

This example shows how to set and delete a message property.

Example 6-5 MQDLTMQ C language example

```
/* Define data structures and initialize with default values. */
```

```
MQLONG  CompCode    = MQCC_OK;
MQLONG  Reason      = MQRC_NONE;
MQCMHO  CrtMsgHOpts = {MQCMHO_DEFAULT};
MQHMSG  hMsg        = MQHM_UNUSABLE_HMSG;
MQPD    PropDesc    = {MQPD_DEFAULT};
MQSMPO  SetPropOpts = {MQSMPO_DEFAULT};
MQDMPO  DltPropOpts = {MQDMPO_DEFAULT};
MQCHARV Name        = {MQCHARV_DEFAULT};

/* Connect to the queue manager */

MQCONN(“QM”, &hConn, &CompCode, &Reason);

/* Create a new message handle */

MQCRTMH(hConn, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);

Name.VSPtr = “my.sport”;
Name.VSLength = MQVS_NULL_TERMINATED;

/* Set a null-terminated property value */

MQSETMP( hConn, hMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_STRING,
         MQVL_NULL_TERMINATED, “football”, &CompCode, &Reason );

/* Delete the property just set */

MQDLTMP(hConn, hMsg, &DltPropOpts, &Name, &CompCode, &Reason);
```

6.2.8 MQBUFMH converts buffer into message handle

This function call converts a WebSphere MQ V7.0 message into a message handle. The inverse function call is MQMHBUF.

This is an advanced Message Queue Interface call therefore it is not used by most application programmers.

This function call causes parsing of MQMD and MQRFH2 headers in the message and converts them to properties. The MQRFH2 properties are removed from the header and the message format is adjusted to represent the message in the buffer.

Syntax

This is the MQBUFMH function syntax:

```
MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason)
```

Parameters

Table 6-7 MQBUFMH parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the Queue manager.
Hmsg	MQHMSG	Input	Message handle for which the buffer is required.
BufMsgHOpts	MQBMHO	Input	Options to control how the message is converted.
MsgDesc	MQMD	Input/Output	On input, it describes the message on the buffer. The CCSID and encoding values must describe the data in the buffer. On output, it describes the message with the MQRFH2 properties removed.
BufferLength	MQLONG	Input	Length in bytes of the Buffer.
Buffer	MQBYTESx	Input/Output	On input this is the message to be converted to Message Handle. On output this is the message with the properties removed.
DataLength	MQLONG	Output	This is the length in bytes of the buffer with the properties removed.
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

This example shows how to convert a message buffer into a message handle.

Example 6-6 MQBUFMH C Language example

```
/* Define data structures and initialize with default values. */

MQHCONN hConn       = MQHC_UNUSABLE_HCONN;
MQLONG  CompCode    = MQCC_OK;
MQLONG  Reason      = MQRC_NONE;
MQCMHO  CrtMsgHOpts = {MQCMHO_DEFAULT};
```

```
MQHMSG  hMsg        = MQHM_UNUSABLE_HMSG;
MQBMHO  BufMsgHOpts = {MQBMHO_DEFAULT};
MQBYTE  Buffer[4096];
MQMD    MsgDesc     = {MQMD_DEFAULT};
MQOD    ObjDesc     = {MQOD_DEFAULT};
MQHOBJ  hObj        = MQHO_UNUSABLE_HOBJ;
MQGMO   GetMsgOpts  = {MQGMO_DEFAULT};
MQLONG  DataLength;
MQLONG  OutputBufferLength;

/* Connect to the queue manager */

MQCONN("QM", &hConn, &CompCode, &Reason);

/* Open a queue */

memcpy(ObjDesc.ObjectName, "Q", MQ_Q_NAME_LENGTH);
MQOPEN(hConn, &ObjDesc, MQOO_INPUT_SHARED, &hObj, &CompCode, &Reason);


/* Get a message from the queue */

MQGET( hConn, hObj, &MsgDesc, &GetMsgOpts, sizeof(Buffer), Buffer,
      &DataLength, &CompCode, &Reason);


/* Create a new message handle */

MQCRTMH(hConn, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);

/* Convert the buffer to a message handle */

MQBUFMH( hConn, hMsg, &BufMsgHOpts, &MsgDesc, DataLength, Buffer,
         &OutputBufferLength, &CompCode, &Reason );
```

6.2.9 MQMHBUF convert a message handle into a buffer

This function call converts a message handle into a buffer. MQBUFMH is the inverse function.

This is an advanced Message Queue Interface call that can be used, for example, by MQGET API exits when the exits want to access certain properties using message property APIs, but they want to pass a buffer back to the application that is designed to cope with MQRFH2 headers rather than message handles.

Syntax

This is the MQMHBUF function syntax:

```
MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason)
```

Parameters

These are the MQMHBUF parameters:

Table 6-8 MQMHBUF parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the Queue manager.
Hmsg	MQHMSG	Input	Message handle to be converted to buffer.
MsgHBufOpts	MQMBHO	Input	Options to control how the message handle is converted.
Name	MQCHARV	Input	The name of the property or properties to put in the buffer. Wildcard character "%" may be used at the end of the property name to put many properties in the buffer.
MsgDesc	MQMD	Input/Output	On input it describes the contents of the buffer area. On output the encoding, CCSID and format reflect the data returned in the buffer.
BufferLength	MQLONG	Input	Length in bytes of the Buffer.
Buffer	MQBYTESx	Output	This is the area to contain the message with the message properties in the MQRFH2.
DataLength	MQLONG	Output	This is the length of the message returned in Buffer including the properties.
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

This example shows how to convert a message handle into a message buffer.

Example 6-7 MQMHBUF C language example

```
/* Define data structures and initialize with default values. */

MQHCONN hConn       = MQHC_UNUSABLE_HCONN;
MQLONG  CompCode    = MQCC_OK;
MQLONG  Reason      = MQRC_NONE;
MQCMHO  CrtMsgHOpts = {MQCMHO_DEFAULT};
MQHMSG  hMsg        = MQHM_UNUSABLE_HMSG;
MQPD    PropDesc    = {MQPD_DEFAULT};
MQSMPO  SetPropOpts = {MQSMPO_DEFAULT};
MQCHARV Name        = {MQCHARV_DEFAULT};
MQMHBO  MsgHBufOpts = {MQMHBO_DEFAULT};
MQMD    MsgDesc     = {MQMD_DEFAULT};
MQBYTE  Buffer[4096];
MQLONG  DataLength;


/* Connect to the queue manager */

MQCONN(“QM”, &hConn, &CompCode, &Reason);

/* Create a new message handle */

MQCRTMH(hConn, &CrtMsgHOpts, &hMsg, &CompCode, &Reason);

Name.VSPtr = “Sport”;
Name.VSLength = MQVS_NULL_TERMINATED;

/* Set a null-terminated property value */

MQSETMP( hConn, hMsg, &SetPropOpts, &Name, &PropDesc, MQTYPE_STRING,
         MQVL_NULL_TERMINATED, “football”, &CompCode, &Reason );

/* Convert the message handle to a buffer */

MQMHBUF( hConn, hMsg, &MsgHBufOpts, &MQPROP_INQUIRE_ALL , &MsgDesc,
         sizeof(Buffer), Buffer, &DataLength, &CompCode, &Reason );
```

6.3 Message browsing

Message browsing has been enhanced in WebSphere MQ V7.0. New options in the MQOPEN options and MQGET options resolve some issues and enable new possibilities when browsing queues.

Message browsing is mainly used by dispatcher programs. A dispatcher is a program that browses a queue continuously to monitor for different types of messages arriving to the queue and passing the messages to different consumers according to the message type. Examples of these dispatcher applications are trigger monitors, CICS bridge dispatcher and Message Driven Beans (MDB).

Before WebSphere MQ V7.0 there were some issues when browsing messages:

- Skipping messages on the queue as result of priority or uncommitted messages or rolled back messages.
- Multiple dispatchers stealing messages from each other.

The above issues were caused by the use of the browsing cursor that always moves forward without considering the arrival (or commit) of new high priority messages, therefore messages are not retrieved during the browse loop. The cursor is associated to connection handles therefore two dispatchers browsing the same queue browses the same messages.

WebSphere MQ V7.0 introduces the following features in message browsing:

- Support of message tokens for distributed queue managers as well as z/OS queue managers. z/OS queue managers supported tokens in previous versions of WebSphere MQ.
- Support of browsing with mark to indicate which messages have been browsed.
- Support of cooperative dispatchers to allow multiple programs to browse the same queue without interference between each other.

6.3.1 Message tokens

The message token is a 16-byte unique identification generated by the queue manager and returned after the MQGET function call in the Get Message Options (GMO) data structure in field MsgToken. The message token uniquely identifies a message on a queue. The message token remains with the message until it is permanently removed from the queue or the queue manager is restarted.

The new feature of browse and mark requires message tokens to do selective MQGETs using the match option MQMO_MATCH_MSG_TOKEN.

6.3.2 Browse and mark

Browse and mark are new options in the MQGET function call that enable the queue manager to keep track which messages have been browsed and who has browsed the messages.

Browse and mark resolves the issue of skipping messages during a browse loop when high priority messages arrive and the browse cursor is on a lower priority message. The queue manager returns high priority messages as soon as they arrive to the queue because they can be identified as not been marked.

6.3.3 Cooperative dispatchers

Cooperative dispatchers are programs browsing the same queue and performing the same dispatching service in parallel, therefore they are not interested in messages that have been browsed by other cooperative dispatchers on the same queue.

There is a new MQOPEN option MQOO_COOP that indicate that the program is a cooperative browser for the queue that is being opened. There is also a new MQGET option to browse and mark messages for cooperative dispatchers.

6.3.4 New MQOPEN option

New options available for the MQOPEN function call:

Table 6-9 New MQOPEN option for cooperative browsing

MQOO_option	Description
MQOO_CO_OP	Used normally with MQOO_BROWSE to indicate that cooperative browsing is used for this queue.

6.3.5 New MQGET options

These new message browsing options are used in combination with the following existing options:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT

Table 6-10 New MQGET options

MQGMO_option	Description
MQGMO_MARK_BROWSE_HANDLE	To mark a message as browsed within the scope of this connection handle.
MQGMO_UNMARK_BROWSE_HANDLE	To unmark a previously browsed message. This option is used in combination with MQMO_MATCH_MSG_TOKEN.
MQGMO_MARK_BROWSE_CO_OP	To mark a message as browsed within the scope of all cooperative dispatchers.
MQGMO_UNMARK_BROWSE_CO_OP	To unmark a previously browsed message. This option is used in combination with MQMO_MATCH_MSG_TOKEN.
MQGMO_UNMARKED_BROWSE_MSG	To browse unmarked messages.

6.3.6 New Queue manager attribute

Queue manager attribute MARKINT indicates the mark time interval. When the time interval is expired the mark of the browsed messages is removed and the messages are eligible to be browsed again. This is to prevent that messages remain unconsumed after they have been browsed and marked.

6.3.7 Examples

The description of a simple browser using browse and mark and of a cooperative dispatcher is presented here:

Simple browser

A program wishes to browse all messages on the queue without skipping any messages and when the return code MQRC_NO_MSG_AVAILABLE is received there are not messages on the queue that have not been browsed.

Such a browser program would use the following MQGMO options:

```
MQGMO_BROWSE_FIRST + /*Always browse first to avoid skipping msgs*/
MQGMO_UNMARKED_BROWSE_MSG + /*Browse unmarked messages */
MQGMO_MARK_BROWSE_HANDLE + /*Mark message as browsed */
MQGMO_WAIT /*Wait until messages arrive on the queue */
```

If the browser application decides to consume the message it needs to destructively MQGET the message with the following MQGMO option:

```
MQMO_MATCH_MSG_TOKEN
```

Cooperative dispatcher

A dispatcher program wishes to browse a queue and initiate a consumer based on the content of each message. There may be more that one instance of the dispatcher on the same queue for distribution of the message workload.

Each cooperative dispatcher calls MQOPEN to open the queue with the MQOO_CO_OP option and would browse the queue making repeated MQGET calls with the following MQGMO options:

```
MQGMO_BROWSE_FIRST + /*Always browse first to avoid skipping msgs*/
MQGMO_UNMARKED_BROWSE_MSG + /*Browse unmarked messages */
MQGMO_MARK_BROWSE_CO_OP + /*Mark message as browsed */
MQGMO_WAIT /*Wait until messages arrive on the queue */
```

The dispatcher would start a consumer program, passing the Message Token returned by the MQGET function call when the message is browsed and marked. The consumer program would destructively MQGET the message that matches the message token received.

6.4 Call Back for asynchronous consumers

Call back is a new set of Message Queue Interface function calls that enable consuming messages from multiple queues or topics. This facilitates the implementation of a message driven processing model (other than using message triggering or message dispatchers based on MQGET with wait) or to have programs that are unaware of the message size that is received.

There is a new programming style that could simplify the design and implementation of new programs.

The new programming style is based on registering functions (MQCB function call) that are called back by the queue manager when there are messages available that match the selection criteria. The control function (MQCTL) indicates to the queue manager when to start dispatching the call back functions pass messages to them. The call back functions receive and process messages in asynchronous mode.

This programming style may be used instead of the MQGET function call to receive messages.

6.4.1 Threading modes

Two threading modes can be used depending on the programming language and/or environment where the program is running.

The first threading mode is that the queue manager steals the application program thread to do the call back dispatching and the application program waits (MQCTL with MQOP_START_WAIT) until all the call back functions are deregistered and terminated.

The second threading mode keeps the application program thread after the control function (MQCTL with MQOP_START) is executed and the queue manager allocates new threads to do the call back processing. The application program can continue doing other work while messages are received and processed.

There is an option to control the thread affinity of the call back consumer functions. The MQCTLO_THREAD_AFFINITY option in MQCTL defines that the same thread is used to invoke the consumer functions that use the same connection handle.

6.4.2 Message consumers and Event handlers

Call back can be used to register two types of functions:

- Message consumers.
- Event handlers.

Message consumers

This type of function is called when there is a message available on a queue or a topic and it meets the selection criteria specified.

The following call back sequence describes the life cycle of the consumer:

- Consumer function is registered by MQCB with MQOP_REGISTER option.
- Consumer is started by MQSTAT with MQOP_START option.
- Consumer receive and process messages.
- Consumer is suspended and resumed by MQCTL with MQOP_SUSPEND and MQOP_RESUME options or by MQCB with MQOP_ SUSPEND and MQOP_RESUME options.
- Consumer is stopped by MQCTL with MQOP_STOP option.
- Consumer is deregistered, either explicitly by MQCB with MQOP_DEREGISTER or implicitly, by a MQCLOSE function call.

Event handler

This type of function is called when an asynchronous event is detected by the queue manager that affect the whole call back environment, such as queue manager quiescing or connection stopping.

The following is a call back sequence describing the life cycle of the event handler:

- Event handler function is registered by MQCB with MQOP_REGISTER option.
- Event handler is started by MQCTL with MQOP_START option.
- Event handler receives and processes events.
- Event handler is suspended and resumed by MQCTL with MQOP_SUSPEND and MQOP_RESUME options or by MQCB with MQOP_ SUSPEND and MQOP_RESUME options.
- Event handler is stopped by MQCTL with MQOP_STOP option.
- Event handler is deregistered by MQCB with MQOP_ DEREGISTER.

6.4.3 MQCB manage call back

The MQCB function call registers a call back function for a specified object handle (queue or topic). A call back function is a piece of code (specified as a function that can be dynamically linked or as a function pointer) that is called by the queue manager when a message is available or an event has occurred, depending on the type of consumer.

Syntax

This is the MQCB function syntax:

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,
CompCode, Reason)
```

Parameters

These are the MQCB parameters:

Table 6-11 MQCB parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	This is a handle which represents a valid connection to the queue manager.

Parameter name	Data type	Input/Output	Description
Operation	MQLONG	Input	Describes the operation being processed by MQCB. The following are the supported operations: MQOP_REGISTER MQOP_DEREGISTER MQOP_SUSPEND MQOP_RESUME.
CallbackDesc	MQCBD	Input	Describes the call back function that is registered. This is only required for MQOP_REGISTER.
Hobj	MQHOBJ	Input	Object handle from which messages are consumed (queue or topic). Not required for event handlers.
MsgDesc	MQMD	Input	Defines the attributes of the messages for the message consumer. Not required for event handler.
GetMsgOpts	MQGMO	Input	Options to control how messages are obtained for the message consumer. Not required for event handlers.
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

6.4.4 MQCBD Call back descriptor

This data structure identifies the call back function that is being registered and the options used during the registration.

Table 6-12 MQCBD Call back descriptor

Parameter name	Data type	Input/Output	Description
StrucID	MQCHAR4	Input	Structure identifier
Version	MQLONG	Input	Current version is 1.
CallbackType	MQLONG	Input	Call back function types. There are 2 types: MQCBCT_MESSAGE_CONSUMER MQCBCT_EVENT_HANDLER

Parameter name	Data type	Input/Output	Description
Options	MQLONG	Input	Options to indicate if call back function is called to indicate the following consumer state changes: MQCBDO_FAIL_IF_QUIESCING MQCBDO_REGISTER_CALL MQCBDO_START_CALL MQCBDO_STOP_CALL MQCBDO_DEREGISTER_CALL
MaxMsgLength	MQLONG	Input	This is the length in bytes of the longest message that can be read from the object handle and given to the call back routine. There is a value to indicate that messages should be delivered without truncation: MQCBD_FULL_MSG_LENGTH
CallbackFunction	MQCB_FUNC TION	Input	This is a pointer to the call back function. You must specify either CallbackFunction or CallbackName but not both.
CallbackName	MQCHAR128	Input	Name of the function that is invoked as a dynamically linked program. You must specify either CallbackFunction or CallbackName but not both.
Reserved			This a reserved field for future use.
CallbackArea	MQPTR	Input/Output	Optional field. It can be a pointer to a memory area that can be used by the call function to store state that has affinity to the object handle.

6.4.5 MQCTL Control call backs

This function performs controlling actions on the call back functions. It executes operations like: START, STOP, SUSPEND and RESUME.

There is an operation called START with WAIT that suspends the calling application program until all the call back functions have terminated and closed the object handle of the queue or topic from which messages are consumed.

Syntax

This the MQCTL function syntax:

```
MQCTL (Hconn, Operation, ControlOpts, CompCode, Reason)
```

Parameters

These are MQCTL function parameters:

Table 6-13 MQCTL parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	This is a handle which represents a valid connection to the queue manager.
Operation	MQLONG	Input	Describes the operation being processed by MQCB. The following are the supported operations: MQOP_START MQOP_START_WAIT MQOP_STOP MQOP_SUSPEND MQOP_RESUME
ControlOpts	MQCTLO	Input/Output	Options that control the action of MQCTLO. The following are the values that can be specified in the Options field: MQCTLO_FAIL_IF_QUIESCING MQCTLO_THREAD_AFFINITY
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

The following sample code shows how to register a message consumer call back function to process messages from a queue. This program starts the call back processing and then waits for input from the keyboard to stop call back processing and close the object and connection handles.

Example 6-8 MQCB and MQCTL C language example

```
/*   Declare MQI structures needed                             */
   MQOD     od = {MQOD_DEFAULT};    /* Object Descriptor          */
   MQMD     md = {MQMD_DEFAULT};    /* Message Descriptor         */
   MQGMO   gmo = {MQGMO_DEFAULT};   /* get message options        */
   MQCBD   cbd = {MQCBD_DEFAULT};   /* Callback Descriptor        */
   MQCTLO  ctlo= {MQCTLO_DEFAULT};  /* Control Options            */
      /** note, sample uses defaults where it can **/

   MQHCONN  Hcon = MQHC_UNUSABLE_HCONN;     /* connection handle   */
   MQHOBJ   Hobj = MQHO_UNUSABLE_HOBJ;      /* object handle       */
   MQLONG   O_options;              /* MQOPEN options             */
   MQLONG   C_options;              /* MQCLOSE options             */
   MQLONG   CompCode;               /* completion code             */
   MQLONG   OpenCode;               /* MQOPEN completion code      */
   MQLONG   Reason;                 /* reason code                 */
   MQLONG   CReason;                /* reason code for MQCONN      */
   char     QMName[50];             /* queue manager name          */

   printf("Sample start\n");
   if (argc < 2)
   {
     printf("Required parameter missing - queue name\n");
     printf("Usage: SAMPLE queue name <qmgr name> <get options>\n");
     exit(999);
   }

   /*   Create object descriptor for subject queue        */

   strcpy(od.ObjectName, argv[1]);
   QMName[0] = '\0';   /* default */
   if (argc > 2)
     strcpy(QMName, argv[2]);

    /*   Connect to queue manager      */

   MQCONN(QMName,                  /* queue manager                */
          &Hcon,                   /* connection handle            */
          &CompCode,               /* completion code              */
          &CReason);               /* reason code                  */

   /* report reason and stop if it failed     */
   if (CompCode == MQCC_FAILED)
   {
     printf("MQCONN ended with reason code %d\n", CReason);
     exit( (int)CReason );
   }

   /*   Open the named message queue for input; exclusive or shared  */
   /*   use of the queue is controlled by the queue definition here  */

   if (argc > 3)
   {
     O_options = atoi( argv[3] );
     printf("open  options are %d\n", O_options);
   }
   else
```

```
{
  O_options = MQOO_INPUT_AS_Q_DEF    /* open queue for input      */
            | MQOO_FAIL_IF_QUIESCING /* but not if MQM stopping   */
            ;                        /* = 0x2001 = 8193 decimal   */
}

MQOPEN(Hcon,                       /* connection handle           */
       &od,                        /* object descriptor for queue */
       O_options,                  /* open options                */
       &Hobj,                      /* object handle               */
       &OpenCode,                  /* completion code             */
       &Reason);                   /* reason code                 */

if (OpenCode == MQCC_FAILED)
{
  printf("MQOPEN ended with reason code %d\n", Reason);
  goto MOD_EXIT;
}

/*   Register a consumer                                          */

cbd.CallbackFunction = MessageConsumer;

MQCB(Hcon,
     MQOP_REGISTER,
     &cbd,
     Hobj,
     &md,
     &gmo,
     &CompCode,
     &Reason);
if (CompCode == MQCC_FAILED)
{
  printf("MQCB ended with reason code %d\n", Reason);
  goto MOD_EXIT;
}

/*  Start consumption of messages                                 */

MQCTL(Hcon,
      MQOP_START,
      &ctlo,
      &CompCode,
      &Reason);
if (CompCode == MQCC_FAILED)
```

```
   {
     printf("MQCTL ended with reason code %d\n", Reason);
     goto MOD_EXIT;
   }

   /*  Wait for the user to press enter                             */

   {
     char Buffer[10];
     printf("Press enter to end\n");
     fgets(Buffer,sizeof(Buffer),stdin);
   }

   /*  Stop consumption of messages                                 */

   MQCTL(Hcon,
         MQOP_STOP,
         &ctlo,
         &CompCode,
         &Reason);
   if (CompCode == MQCC_FAILED)
   {
     printf("MQCTL ended with reason code %d\n", Reason);
     goto MOD_EXIT;
   }

MOD_EXIT:

   /*   Close the source queue (if it was opened)                   */

   if (Hobj != MQHO_UNUSABLE_HOBJ)
   {
     if (argc > 4)
     {
       C_options = atoi( argv[4] );
       printf("close options are %d\n", C_options);
     }
     else
     {
       C_options = MQCO_NONE;        /* no close options             */
     }

     MQCLOSE(Hcon,                   /* connection handle            */
             &Hobj,                  /* object handle                */
             C_options,
             &CompCode,              /* completion code          */
             &Reason);               /* reason code              */

                                      /* report reason, if any    */
     if (Reason != MQRC_NONE)
     {
       printf("MQCLOSE ended with reason code %d\n", Reason);
     }
   }

   /*   Disconnect from MQM if not already connected             */

   if (Hcon != MQHC_UNUSABLE_HCONN)
   {
     if (CReason != MQRC_ALREADY_CONNECTED )
     {
       MQDISC(&Hcon,                  /* connection handle        */
              &CompCode,              /* completion code          */
              &Reason);               /* reason code              */

       /* report reason, if any     */
       if (Reason != MQRC_NONE)
       {
         printf("MQDISC ended with reason code %d\n", Reason);
       }
     }
   }

   /* END OF SAMPLE                                               */

   printf("Sample end\n");
   return((int)Reason);
```

6.4.6 Call Back function

This is the function that is registered and called when messages are available on the associated object handle or events have been detected.

A message consumer function is free to execute any Message Queue Interface calls with the exception of MQCTL with operation MQOP_START_WAIT, and MQDISC.

Only one call back function is allowed to be registered per object handle using the same queue manager connection handle. If a second call back function is registered for the same object handle and same connection handle it replaces the previous registration.

Messages from the same object handle and same connection handle (Hconn) are processed one at a time preserving the order specified (priority or first in first out) in the MQGMO options in the MQCB function call.

If multiple registrations are created to consume messages from multiple object handles (queues or topics) and they share the same queue manager connection handle the messages are processed one at a time. A queue manager connection handle cannot have more than one Message Queue Interface call active at the same time.

It is possible to register multiple call back functions for the same object handle using different queue manager connection handles. In this case messages are processed in parallel and the sequence cannot be guaranteed.

Syntax

This is the call back function signature:

```
MQLONG MyCBFunction (Hconn, MsgDesc, GetMsgOpts, Buffer, Context)
```

Parameters

These are the parameters received by the call back function:

Table 6-14 Call back function input parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	This is a handle which represents a valid connection to the queue manager.
MsgDesc	MQMD	Input	Defines the attributes of the messages for the message consumer. Not required for event handler.
GetMsgOpts	MQGMO	Input	Options to control how messages are obtained for the message consumer. Not required for event handlers.
Buffer	MQBYTEx	Input	Area containing the message.
Context	MQCBC	Input/Output	Data structure with context information for Call Back functions.

MQCBC Call back context

This is data structure is passed to call back functions as the context information.

Table 6-15 MQCBC Call back context

Field name	Data type	Input/Output	Description
StrucID	MQCHAR4	Input	Structure identifier
Version	MQLONG	Input	Current version is 1.
CallType	MQLONG	Input	Information about why this function was called. There are 7 call types: MQCBCT_REGISTER_CALL MQCBCT_START_CALL MQCBCT_STOP_CALL MQCBCT_DEREGISTER_CALL MQCBCT_MSG_REMOVED MQCBCT_MSG_NOT_REMOVED MQCBCT_EVENT_CALL
Hobj	MQHOBJ	Input	Object handle from which messages are consumed.
CompCode	MQLONG	Input	Completion code. It indicates if there were problems retrieving the message.
Reason	MQLONG	Input	Reason code.
State	MQLONG	Input	Indication of the state of the current consumer. This field is of most value when a non-zero reason code is passed.
DataLength	MQLONG	Input	Length in bytes of the message.
BufferLength	MQLONG	Input	Length in bytes of the buffer used to store the message.
Flags	MQLONG	Output	Flags containing information about this consumer. The following flag may be returned: MQCBCF_BUFF_EMPTY when there has been an error during the read ahead process.
CallbackArea	MQPTR	Input/Output	Field available for the call back function to use. This area is saved and returned unchanged by the queue manager. It can be used by the call back to store state information that is preserved across invocations of the function. This area is shared by call back function invocations with this object handle.

Field name	Data type	Input/Output	Description
ConnectionArea	MQPTR	Input/Output	Field available for the call back function to use. This area is saved and returned unchanged by the queue manager. It can be used by the call back to store state information that is preserved across invocations of the function. This area is shared for all call back functions that have the same connection handle.

Example

This call back function prints a message for each type of call back received.

Example 6-9 Call function example

```
/****************************************************************/
/* FUNCTION: MessageConsumer                                     */
/* PURPOSE : Callback function called when messages arrive */
/****************************************************************/
 MQLONG MessageConsumer(MQHCONN   hConn,
                        MQMD    * pMsgDesc,
                        MQGMO   * pGetMsgOpts,
                        MQBYTE  * Buffer,
                        MQCBC   * pContext)
 {
   MQLONG i,max;

   switch(pContext->CallType)
   {
     case MQCBCT_REGISTER_CALL:
     case MQCBCT_START_CALL:
     case MQCBCT_STOP_CALL:
     case MQCBCT_DEREGISTER_CALL:
     printf("Calltype = %d\n",pContext->CallType);
          break;

     case MQCBCT_MSG_REMOVED:
     case MQCBCT_MSG_NOT_REMOVED:

     printf("Calltype = %d\n",pContext->CallType);
          printf("Message received\n");
     break;

     case MQCBCT_EVENT_CALL:
          printf("Event Call : Reason = %d\n",pContext->Reason);
```

```
        break;

    default:
        printf("Calltype = %d\n",pContext->CallType);
        break;
    }
    return 0;
}
```

6.5 Publish/Subscribe

The Message Queue Interface has been extended to support Publish/Subscribe in WebSphere MQ V7.0.

In Pub/Sub there are two roles that use different Message Queue Interface function calls and data structures. The roles are the Publisher and the Subscriber.

Topics are the strings that define the destination of the messages produced by Publishers and the source of the messages consumed by the Subscribers.

6.5.1 Topics

The data structure Object Descriptor (MQOD) has been extended to support the use of topics when the MQOPEN or MQPUT1 functions calls are used by publisher programs.

There is a new data structure Subscription Descriptor (MQSD) that is used to specify the topic and the subscription attributes. A subscriber application uses the MQSUB function call with a reference to MQSD to register a subscription to receive publication messages.

Topics can be defined in three ways:

- Using a topic object that is created using queue manager administration commands. This object has a 48 character topic object name and a variable length topic string as an attribute. The topic object name may be used in the MQOD or MQSD data structures as a short name reference to a topic.
- Using a topic string in the MQOD or the MQSD data structures. Topic strings are variable length strings (MQCHARV) of up to 10240 bytes long.

- Using a concatenation of the topic string defined in a topic object and the topic string defined in the MQOD or MQSD data structures. The concatenation is performed when the topic object name and the topic string are both specified in the MQOD or MQSD.

6.5.2 Publishers

Publisher applications are programs that put messages to topic destinations. In earlier versions of WebSphere MQ, publishers have to put messages on queues and these messages had to be formatted with special headers that were required by the Pub/Sub brokers.

Publisher applications have the option to suppress any reply information that might be present in the MQMD of the message. Pub/Sub does not normally expect that subscribers reply to publication messages.

Publisher applications can publish messages as retained publications. Retained publications are kept by the queue manager and delivered to new subscribers when they subscribe to the topic and request to receive retained publications. Only one retained publication per topic is kept. A new retained publication replaces the previous one.

An application program needs the MQOPEN, MQPUT and MQCLOSE or MQPUT1 function calls to publish messages. Changes have been made to MQMD, MQOD and MQPMO data structures to support the publication to topics. See Example 6-10 on page 131.

6.5.3 Subscribers

To support subscriber applications, WebSphere MQ V7.0 has two new Message Queue Interface function calls (MQSUB and MQSUBRQ), new data structures (MQSD and MQSRO) and changes to existing data structures (MQMD).

A subscriber application needs to perform at least two actions: register a subscription and receive publications.

There are two types of subscriptions:

- Durable: These subscriptions remain active after the subscriber program closes the connection to the queue manager. All publications are stored in the queue manager while the subscription is suspended. The subscriber is expected to reconnect and resume the subscription to receive all publications.
- Non durable: These subscriptions are terminated when the subscriber program terminates and/or closes the connection to the queue manager.

There are two modes of storing publications for the subscribers:

- Managed: These are subscriptions that the queue manager provides the queue to store the publications before the subscriber receives them. It is recommended that managed subscriptions are also non-durable. If this is the case the queue manager removes from the queue all the unconsumed messages when the subscription is terminated.
- Non managed: These are subscriptions that the subscriber application needs to provide a queue to store the publications before they are received by the subscriber. It is recommended that non managed subscriptions are also durable. In this case it is a responsibility of the subscriber application program to consume or clear all messages from the queue.

Subscribers can request to receive retained publications when the subscription is registered. The MQSUBRQ function call enables a subscriber to request retained publications on demand.

Subscribers receive publications using MQGET or MQCB function calls. If the subscription is non managed then the subscriber can open the subscriber queue and use MQGET to retrieve the publications without making a reference to a topic. This is useful when non-managed subscriptions have been created using administration commands and non Pub/Sub applications can receive publications. Example 6-11 on page 132.

6.5.4 MQOD Object descriptor

The following are the new fields added to the Version 4 of the MQOD Object descriptor:

Table 6-16 MQOD version 4 new fields

Field name	Data type	Input/Output	Description
...version 3 fields...			Version 3 fields precede the following new fields in the Version 4 MQOD.
ObjectString	MQCHARV	Input	Optional. Topic string to be used for publications. If field ObjectName is specified then the topic is a concatenation of the topic string in the topic object and the string specified here.
SelectionString	MQCHARV	Input	Optional. This field is used to specify a selection string to be used by MQGET calls.

Field name	Data type	Input/Output	Description
ResObjectString	MQCHARV	Output	This field contains the topic string from the topic object specified in ObjectName and the ObjectString is not specified or the result of the concatenation of the topic string in the topic object and the ObjectString.
ResolvedType	MQLONG	Output	If ObjectName specified a queue alias then this field has the alias object type (queue or topic).

6.5.5 MQSD Subscription descriptor

The following is the new subscription description data structure used by the MQSUB function call:

Table 6-17 MQSD Subscription descriptor fields

Field name	Data type	Input/Output	Description
StrucId	MQCHAR4	Input	Structure Identifier.
Version	MQLONG	Input	Version 1.
Options	MQLONG	Input	At least one the following options: MQSO_ALTER MQSO_CREATE MQSO_RESUME MQSO_CREATE + MQSO_RESUME MQSO_CREATE + MQSO_ALTER
ObjectName	MQCHAR48	Input	Topic object name as defined in the queue manager (optional). It can be used instead of or in combination with ObjectString.
AlternateUserId	MQCHAR12	Input	Alternate userid to check authorization for this subscription.
AlternateSecurityID	MQBYTE40	Input	Security identifier passed with the AlternateUserid.
SubExpiry	MQLONG	Input/Output	Time specified in tenths of a second after which the subscription expires. On return of a MQSO_RESUME this is the original expiry time of the subscription and not the remaining expiry time.
ObjectString	MQCHARV	Input	Topic string. It can be used instead of or in combination with ObjectName.

Field name	Data type	Input/Output	Description
SubName	MQCHARV	Input	Unique subscription name required when a durable subscription is used.
SubUserData	MQCHARV	Input	User data that is included with every publication received with this subscription. The user data is included as MQSubUserData message property.
SubCorrelId	MQBYTE24	Input/Output	This Correlation Id is set in the MQMD of all messages matching this subscription.
PubPriority	MQLONG	Input/Output	This priority is in the MQMD of all messages matching this subscription.
PubAccountingToken	MQBYTE32	Input/Output	This Accounting Token is in the MQMD of all messages matching this subscription.
PubApplIdentityData	MQCHAR32	Input/Output	This Application Identity Data is in the MQMD of all messages matching this subscription.
SelectionString	MQCHARV	Input/Output	This is the selection string used to select publications matching this topic. On return of a MQSUB call with MQSO_RESUME this field has the original selection string if a buffer long enough has been provided.
SubLevel	MQLONG	Input/Output	Subscription level to be used for this subscription. This field is used by intercepting subscribers and not by common Pub/Sub applications.
ResObjectString	MQCHARV	Output	This field contains the topic string from the topic object specified in ObjectName and if ObjectString is not specified or the result of the concatenation of the topic string in the topic object and the ObjectString

6.5.6 MQPMO Put-message options

The following table shows the fields that have new uses or new fields that have been added in WebSphere MQ V7.0:

Table 6-18 MQPMO version 3 field changes

Field name	Data type	Input/Output	Description
StrucId	MQCHAR4	Input	Structure Identifier.

Field name	Data type	Input/Output	Description
Version	MQLONG	Input	Version 3.
Options	MQLONG	Input	New options for publication: MQPMO_SUPPRESS_REPLYTO MQPMO_RETAIN
... other fields ...			Other existing fields precede the following:
ResolvedQName	MQCHAR48	Output	This field is returned with an undefined value for topics.
ResolvedQMgrName	MQCHAR48	Output	This fields is returned with an undefined value for topics.
--- Other fields ...			Other fields precede the following fields added in version 3 of MQPMO:
OriginalMsgHandle	MQHMSG	Input	Original message handle received by this program and this field is used with Action MQACTP_REPLY or MQACTP_FORWARD.
NewMsgHandle	MQHMSG	Input/Output	New message handle with message properties that overrides the Original Message Handle.
Action	MQLONG	Input	Field to specify the type of put being performed and the relationship with a previous message received (original message). The action values are: MQACTP_NEW MQACTP_FORWARD MQACTP_REPLY MQACTP_REPORT
PubLevel	MQLONG	Input	This is the level of subscription targeted by this publication. This field is used by intercepting subscribers and not by common Pub/Sub applications.

6.5.7 MQMD Message description

The following are the changes to the message descriptor (MQMD):

Table 6-19 MQMD changes

Field name	Data type	Input/Output	Description
StrucId	MQCHAR4	Input	Structure Identifier.
Version	MQLONG	Input	Version 2.
Report	MQLONG	Input	If reports are requested by a publisher then this application should expect zero, one or many report messages back, depending on how many subscribers got the publication.
... other fields ...			Other existing fields precede the following:
Priority	MQLONG	Input/Output	New value that can be specified: MQPRI_PRIORITY_AS_TOPIC_DEF
... other fields ...			Other existing fields precede the following:
MsgId	MQBYTE24	Input	On MQPUT/MQPUT1 for a retained publication this field has the MsgId used for the retained publication.
CorrelId	MQBYTE24	Input/Output	On MQPUT or MQPUT1 this can be the CorrelId for retained publications and it can be propagated to all the subscriptions that do not specify a CorrelId in the subscription.
... other fields ...			Other existing fields precede the following:
UserIdentifier	MQCHAR12		Used for retained publications but not used for other types of publications.
AccountingToken	MQBYTE32		Used for retained publications but not used for other types of publications.
ApplIdentityData	MQCHAR32		Used for retained publications but not used for other types of publications.
PutAppType	MQLONG		This value can be overridden by the value specified in the subscription.
PutAppName	MQCHAR28		This value can be overridden by the value specified in the subscription.
PutDate	MQCHAR8		This value can be overridden by the value specified in the subscription.

Field name	Data type	Input/Output	Description
PutTime	MQCHAR8		This value can be overridden by the value specified in the subscription.
ApplOriginData	MQCHAR4		This value can be overridden by the value specified in the subscription.

6.5.8 MQSUB Manage subscription

This is a new function call to register, alter or deregister a subscription to a topic.

The topic can be specified by a topic object or a topic string or a concatenation of the topic object and the topic string. See Example 6-11 on page 132.

Syntax

This is the MQSUB function syntax:

```
MQSUB ( Hconn, SubDesc, Hobj, Hsub, CompCode, Reason )
```

Parameters

These are the MQSUB parameters:

Table 6-20 MQSUB parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Connection handle that represents a valid connection to the queue manager.
SubDesc	MQSD	Input/Output	This is the subscription descriptor that represents the subscription that is registered.
Hobj	MQHOBJ	Input/Output	On input, for non managed subscriptions this is the handle that represents the queue to receive the publications. On output, for managed subscriptions this represents the queue object assigned by the queue manager to receive publications.
Hsub	MQHSUB	Output	This is subscription handle represents the subscription that has been created. Hsub can be used for two further operations: MQSUBRQ to request retained publications or MQCLOSE to remove the subscription.
CompCode	MQLONG	Output	Completion code

Parameter name	Data type	Input/Output	Description
Reason	MQLONG	Output	Reason code

6.5.9 MQSUBRQ Subscription request

This function call requests copies of the current retained publications for the topic. If the subscription is made to a topic with wild cards then this request can generate multiple messages returned to the subscriber.

Syntax

This is the MQSUBRQ function syntax:

```
MQSUBRQ ( Hconn, Hsub, Action, SubRqOpts, CompCode, Reason )
```

Parameters

These are the MQSUBRQ parameters:

Table 6-21 MQSUBRQ parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Connection handle that represents a valid connection to the queue manager.
Hsub	MQHSUB	Input	This is subscription handle represents the subscription
Action	MQLONG	Input	This parameter control the action required with the subscription: MQSR_ACTION_PUBLICATION This request a copy of the retained publications.
SubRqOpts	MQSRO	Input/Output	Options to control the action of MQSUBRQ. There is a field NumPubs that indicates the number of publication messages returned.
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

6.5.10 MQCLOSE Close object

This function call is used to release control of a queue or a topic or a subscription.

The MQCLOSE options may specify if durable subscriptions are kept or removed.

When an object handle of a managed subscription is closed the queue manager removes all publications which have not been retrieved.

When closing non managed subscriptions publications are not removed. It is recommended that subscriber applications process all the messages from the subscription queue after the subscription is closed.

Syntax

This is the MQCLOSE function syntax:

```
MQCLOSE ( Hconn, Hobj, Options, CompCode, Reason )
```

Parameters

These are the MQCLOSE parameters:

Table 6-22 MQCLOSE parameters

Parameter name	Data type	Input/Output	Description
Hconn	MQHCONN	Input	Connection handle that represents a valid connection to the queue manager.
Hobj	MQHOBJ	Input/Output	This handle represents the object to be closed. Objects can be queues or topics or subscriptions. If it is a subscription the Hsub handle returned from MQSUB is used instead of Hobj. On output a unusable handle is returned.
Options	MQLONG	Input	These new options control how subscriptions are closed: MQCO_REMOVE_SUB MQCO_KEEP_SUB (durable subscriptions)
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Examples

The following are two examples of a simple publisher and a simple subscriber programs.

Example 6-10 Simple publisher program

```
/* Define data structures */
```

```
MQHCONN hConn;
MQOD   ObjDesc  = {MQOD_DEFAULT};
MQHOBJ hObj;
MQLONG CompCode, Reason;
MQLONG OpenOpts = 0;
MQPMO pmo;
MQMD MsgDesc;
char pBuffer[ ] = "x";

/* Open a topic to publish to  */

ObjDesc.ObjectType             = MQOT_TOPIC;
ObjDesc.Version                = MQOD_VERSION_4;
ObjDesc.ObjectString.VSPtr     = argv[1];   /* Topic string */
ObjDesc.ObjectString.VSLength  = MQVS_NULL_TERMINATED;

OpenOpts = MQOO_OUTPUT
         | MQOO_FAIL_IF_QUIESCING;

MQOPEN(hConn, &ObjDesc, OpenOpts, &hObj, &CompCode, &Reason);

/* Publish to the specified topic string                         */

MQPUT(hConn, hObj, &MsgDesc, &pmo, strlen(pBuffer), pBuffer, &CompCode, &Reason);
```

The following is the simple subscriber program:

Example 6-11 Simple Subscriber program

```
/* Define data structures */

MQSD   SubDesc  = {MQSD_DEFAULT};
MQHOBJ Hobj     = MQHO_UNUSABLE_HOBJ;
MQHOBJ Hsub     = MQHO_UNUSABLE_HOBJ;
MQHCONN hConn;
MQOD   ObjDesc  = {MQOD_DEFAULT};
MQLONG CompCode, Reason;
MQGMO gmo;
MQMD MsgDesc;
MQLONG DataLength;
char pBuffer[ ] = "x";
```

```
/* Subscribe for publications from a topic */

 SubDesc.ObjectString.VSPtr    = argv[1];  /* Topic string */
 SubDesc.ObjectString.VSLength = MQVS_NULL_TERMINATED;
 SubDesc.SubName.VSPtr         = argv[2];  /* Subscription name */
 SubDesc.SubName.VSLength      = MQVS_NULL_TERMINATED;
 SubDesc.Options               = MQSO_CREATE
                               | MQSO_MANAGED
                               | MQSO_NON_DURABLE
                               | MQSO_FAIL_IF_QUIESCING;

 MQSUB(hConn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason);


/* Consume messages published to the topic  */

MQGET(hConn, Hobj, &MsgDesc, &gmo, strlen(pBuffer), pBuffer, &DataLength,
      &CompCode, &Reason);


/* Unsubscribe  */

if (Hsub != MQHO_UNUSABLE_HOBJ)
{
  MQCLOSE(hConn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
}

/* And close the object handle. */

if (Hobj != MQHO_UNUSABLE_HOBJ)
{
  MQCLOSE(hConn, &Hobj, MQCO_NONE, &CompCode, &Reason);
}
```

6.6 Put action indicators

In WebSphere MQ V7.0 it is now possible to indicate to the queue manager the type of put action that is performed and the relationship between the new message and a possible original message that may have been received before.

The queue manager uses the action indicators to validate, copy message properties and MQMD values from the original message to the new message handle according to the action performed.

The original message is represented by a message handle stored in MQPMO in the OriginalMsgHandle field and the new message is represented by a message handle stored in the NewMsgHandle field.

The type of actions that can be specified in the new Action field in the MQPMO put options in the MQPUT or MQPUT1 function calls are:

- MQACTP_NEW: A new message that is unrelated to any previous message is been put to the queue.
- MQACTP_FORWARD: This is a previously retrieved message that may have been modified is been put on the queue for forward processing.
- MQACTP_REPLY: The new message is a reply to a previously retrieved message.
- MQACTP_REPORT: A report message has been generated as result of receiving a message.

6.7 Message selectors

A message selector is a criteria to filter messages during MQGET or MQCB function calls. The selection criteria is specified during the MQOPEN or MQSUB function calls. To change the selection criteria the object or subscription has to be closed and opened again.

Message selectors existed in Java Message Service (JMS) prior to WebSphere MQ V7.0. Now any application program that uses Message Queue Interface functions calls can use message selectors.

Message selectors act on message properties defined in a message. Fields in the MQMD data structure can be considered as message properties in message selectors.

A message selector is a variable length string (MQCHARV) field in MQOD or MQSD data structures. The syntax of the message selector string is based on a subset of the SQL92 conditional expressions.

When message selectors are used with point to point messaging applications the selection string is specified in the field SelectionString in the MQOD that is a parameter of the MQOPEN function call. Any subsequent MQGET function calls for the object handle returned in the MQOPEN filters the messages based on the selection criteria described in the selection string.

When message selectors are used with Pub/Sub applications the selection string is specified in the field SelectionString in the MQSD that is a parameter of the MQSUB function call. Any subsequent MQGET function calls for the object handle returned in the MQSUB filters the publications based on the selection criteria described in the selection string.

A key requirement for message selection to work is that message producer applications (point to point or publishers) set the message properties required by the message selectors.

Note to Reviewer: An example for message selectors should be included here. The scenario as it is today (14 Dec 2007) does not use message selectors or message properties.

6.8 Other MQI considerations

- MQINQ and MQSET function calls have been updated to inquire and set the new MQ objects and new attributes on existing or new objects.
- API exits support some of the new Message Queue Interface function calls in WebSphere MQ V7.0. Some function calls are not supported because it does not make any sense to have API exits for those calls.
- The equivalent object oriented version of publish/subscribe APIs for WebSphere MQ V7.0 (C++, Base Java and .NET) are out of the scope of this Redbook publication. Refer to the WebSphere MQ V7.0 manuals: *Using C++*, *Using Java* and *Using .Net* for detailed information.

7

WebSphere MQ Java Message Service enhancements

This chapter discusses the enrichments done to the WebSphere MQ implementation for JMS. These enhancements enables WebSphere MQ to be a natural fit for Java Message Service (JMS).

The following topics are discussed:

- "Read ahead"
- "Asynchronous put"
- "Asynchronous consume"
- "Conversation sharing sessions"
- "Mapping of WebSphere MQ and JMS message"
- "Properties of WebSphere MQ classes for Java Message Service"

7.1 Read ahead

Read ahead or Streaming feature in WebSphere MQ V7.0 allows messages from destination to be sent to the Java Message Service (JMS) client ahead of the application actually requesting for the messages. This saves a client from having to send a separate request to the WebSphere MQ server for each message it consumes and allows the client to receive messages as continuous stream of messages.

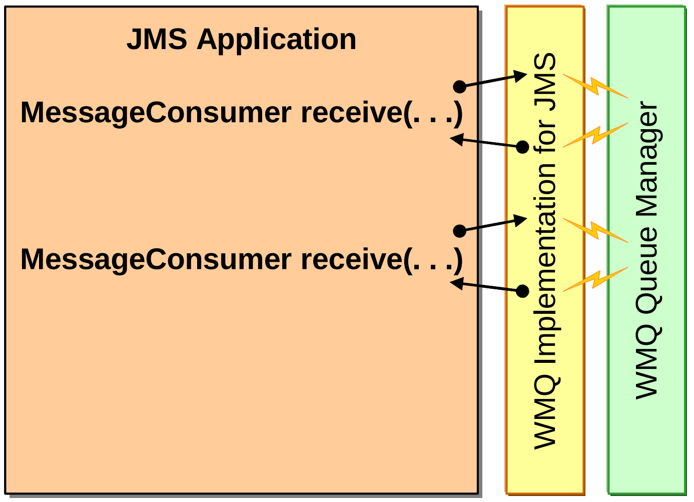
A JMS application uses MessageConsumer to receive messages from the destination. The JMS application can use one of the `receive` methods from the MessageConsumer to fetch messages synchronously from the destination. The JMS application can also register an object that implements JMS MessageListener interface to fetch messages Asynchronously. JMS client consumes both persistent and non-persistent messages from the destination either synchronously or asynchronously. When the JMS client consumes non-persistent messages, the destination on WebSphere MQ server can be configured such that the WebSphere MQ implementation for JMS can make use of an internal buffer to store the messages of interest before delivering them to the application. The concept of storing the messages in the internal buffer before the application actually requests for the messages is the Read ahead feature.

Note: MessageConsumer, MessageListener and receive() method are JMS concepts, for more information on these topics refer *JMS specifications*, available at:

`http://java.sun.com/products/jms/docs.html`

Read ahead feature is applicable only for non-persistent messages and does not apply to persistent messages, because, if persistent messages were read into a buffer, the queue manager would no longer be able to recover the messages in the event of a failure. However, an application that consumes messages from a destination with a mixture of persistent and non-persistent messages can still use Read ahead. The order of the messages is preserved, but the runtime benefits of Read ahead apply only to the non-persistent messages. Read ahead is particularly effective for destinations with a large number of messages that need to be consumed rapidly.

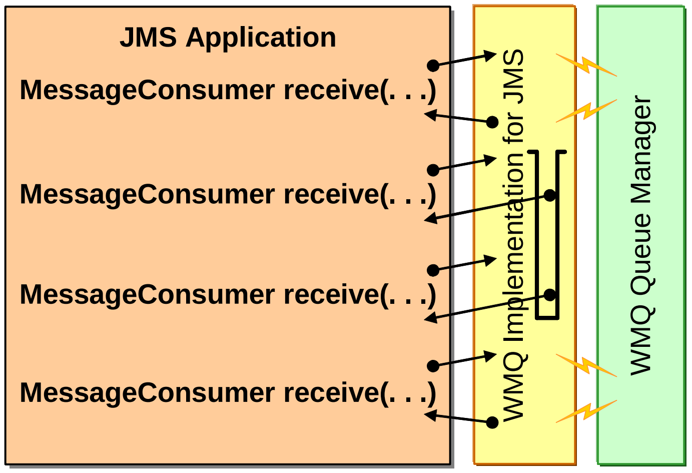
The figure below shows a JMS application receiving messages from a WebSphere MQ V6.0 queue manager. The JMS application has to issue a GET for every message it has to retreive from the queue.


JMS Application
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
WMQ Implementation for JMS
WMQ Queue Manager

- Each message request flows over the network to the queue manager.

The message is then returned over network and passed to the application.

The figure below shows a JMS application retreiving messages from a WebSphere MQ V7.0 queue manager using the Read ahead feature.


JMS Application
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
WMQ Implementation for JMS
WMQ Queue Manager

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

Figure 7-1 JMS application getting messages from WMQ V7.0 queue manager using Read ahead feature.

The following points discusses the considerations for using Read ahead feature in the application design:

- If the JMS application using Read ahead feature terminates abruptly before it consumes all the messages from the internal buffer, then any non-persistent messages currently stored in the buffer are discarded and all those messages are lost.

- Read ahead feature is applicable only for non-persistent messages.
- Any existing JMS application can make use of the Read ahead feature without any JMS client code modifications.
- If all the following conditions are true, messages sent to a queue in a session might not be received in the order in which they were sent:
  - An application uses two message consumers in the same session to consume the messages from the queue.
  - Each message consumer uses a different Destination object for the queue.
  - Any or both of the Destination objects are configured for read ahead.

Note: An attribute READAHEADALLOWED must be set on the destination for the JMS client application to make use of the Read ahead feature in WebSphere MQ V7.0. Refer the section 5.4, "Read ahead" on page 70.

7.2 Asynchronous put

A Java Message Service (JMS) client application responsible for capturing information on climatic changes every minute, for example, change in humidity, temperature, dust or air pollution, sends sequence of messages in rapid succession to the destination. The client application does not require any acknowledge or reply back for every message sent. The application is willing to have unacknowledged communication with the server for every message put or published to the destination.

Prior to WebSphere MQ V7.0, a JMS application connecting to a destination in client connection mode was unable to send or publish messages in rapid succession. The WebSphere MQ implementation for JMS had to wait for a reply back from the queue manager for every message sent by the JMS client. Only after receiving the acknowledgement from the WebSphere MQ (WMQ) server the WMQ implementation for JMS could return the control back to the JMS client application. The JMS client application was then able to send a second message and this cycle repeated.

But, with the enhanced Asynchronous put feature in WebSphere MQ V7.0 the JMS application can send messages in rapid succession. The JMS client application can send a message and the WMQ implementation for JMS forwards the message to the destination. Once the message is forwarded, the control is returned back to the JMS client application. The JMS client application can then send the second message immediately without waiting for the first message getting acknowledged.

Note: JMS applications can make use of the Asynchronous put (Fire and forget) feature only when connecting to the WebSphere MQ V7.0 in Client mode connection and not in Bindings mode connection.

In Client mode connection, the application establishes connection to the WebSphere MQ server using sockets over Transmission Control Protocol/Internet Protocol (TCP/IP).

In Binding mode connection, the application establishes connection to the WebSphere MQ server using shared memory. Both the client application and WebSphere MQ server must be running on the same machine.

The existing JMS applications can make use of the WebSphere MQ V7.0 Asynchronous put feature without having to do any client code modifications. The destination on the WMQ server can be configured independently for the JMS applications to use this feature. For more information on configuring the destination for asynchronous put refer to the section 5.5, “Asynchronous put” on page 73.

WebSphere MQ implementation for JMS works in this way for non-persistent messages and also for persistent messages sent in a transacted session.

Note: From the JMS specifications, the term transacted session refers to the case where a session’s commit and rollback methods are used to demarcate a transaction local to the session. In the case where a session’s work is coordinated by an external transaction manager, a session’s commit and rollback methods are not used and the result of a closed session’s work is determined later by the transaction manager.

For messages sent in a transacted session, the JMS application determines whether the WebSphere MQ queue manager has received all the messages safely or not by committing the transaction. When the application commits the transaction, the queue manager acknowledges if all the messages sent in that transaction were successfully received or not.

For the non-persistent messages sent in a non transacted session, the SENDCHECKCOUNT property of the ConnectionFactory object determines how many messages are to be sent before WebSphere MQ implementation for JMS checks that the queue manager has received the messages safely or not.

Note: For ConnectionFactory object attributes refer to the section 7.6, “Properties of WebSphere MQ classes for Java Message Service” on page 149.

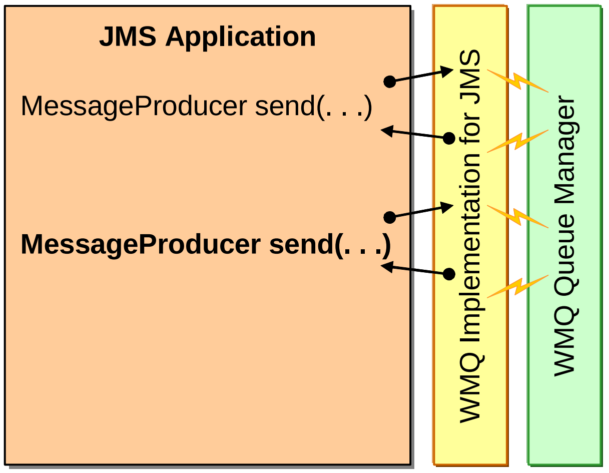
JMS client application may encounter exceptions when sending messages in rapid succession. In such conditions, it is necessary for the JMS client application to register an *ExceptionListener* with the connection object. In the event of failure, that is, when WMQ server determines that one or more messages were not received safely from the application, the WMQ implementation for JMS triggers the onException method of the *ExceptionListener* to pass the JMS exception to the application. The JMS application can be designed to capture the JMS exception and process accordingly.

Note: ExceptionListener is a JMS terminology, for more information on ExceptionListener refer *JMS specifications* available at:

http://java.sun.com/products/jms/docs.html

This optimization is of most benefit to JMS applications that need to send a sequence of messages in rapid succession, but do not require immediate feedback from the queue manager for each message sent.

Figure 7-1 shows the workflow of an JMS application sending messages to in client mode to WebSphere MQ V6.0 queue manager.


JMS Application
MessageProducer send(. . .)
MessageProducer send(. . .)
WMQ Implementation for JMS
WMQ Queue Manager

- Each message passed to the queue manager over the network and stored safely.
- CC and RC are returned after each message is written.

Figure 7-2 JMS sending messages in client mode in WMQ 6.0

Figure 7-2 shows the workflow of an JMS application sending messages to in client mode to WebSphere MQ V7.0 queue manager using Ashynchronous put.

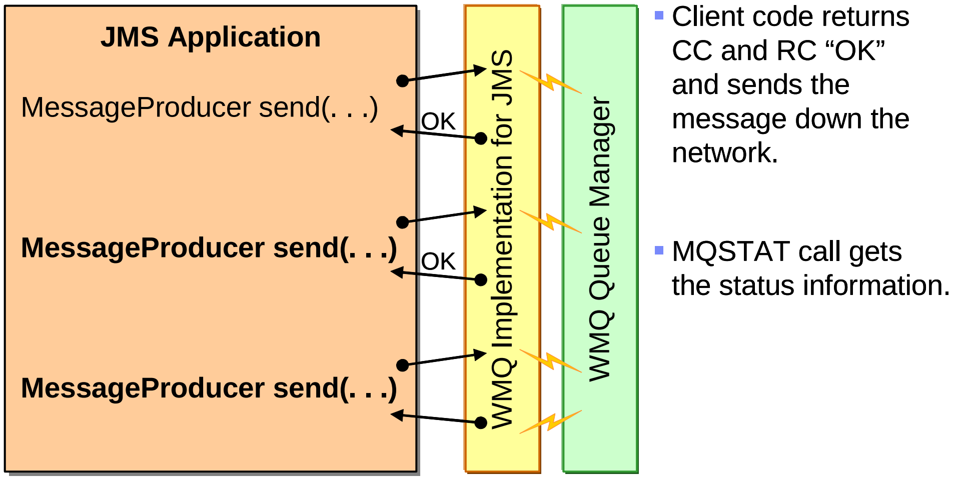

JMS Application
MessageProducer send(. . .)
OK
MessageProducer send(. . .)
OK
MessageProducer send(. . .)
WMQ Implementation for JMS
WMQ Queue Manager
Client code returns CC and RC "OK" and sends the message down the network.
MQSTAT call gets the status information.

Figure 7-3 JMS sending messages in client mode with Asynchronous put

7.3 Asynchronous consume

WebSphere MQ V7.0 supports both synchronous and asynchronous message consumption. When applications need to consume message asynchronously, the application can register a callback function for a destination. When a suitable message is sent to the destination, WebSphere MQ calls the function and passes the message as a parameter. The function can then process the message asynchronously. Note that consuming of messages asynchronously by the Java Message Service (JMS) applications was already present with the previous releases of WebSphere MQ when JMS applications were implementing JMS *MessageListener*. But, with WebSphere MQ V7.0, WebSphere MQ implementation for JMS has been enhanced to take advantage of the callback mechanism available with WebSphere MQ V7.0.

From the application design or JMS client programming perspective there are no changes for consuming the messages asynchronously. The JMS application has to register an object implementing the JMS *MessageListener* interface to consume messages asynchronously. The core enhancement has been done at the WebSphere MQ classes for JMS. This section outlines the change in behavior on how the messages are consumed asynchronously.

The implementation of JMS message listeners is now a more natural fit with WebSphere MQ. In the previous version of WebSphere MQ, the WebSphere MQ implementation for JMS used to poll the destination at regular interval to check if any suitable messages were available on the destination. Polling for messages added an additional workload at the client side in terms of resource usage. In client connection mode, WebSphere MQ classes for JMS had to send polling requests to the WebSphere MQ server over theTransmission Control Protocol/Internet Protocol. This resulted in unnecessary usage of Input/Output (I/O) resources and also the Central Processing Unit (CPU) usage was high even though there were no suitable messages available on the destination. This was because the JMS client application had to poll the destination at regular intervals checking for suitable messages.

In WebSphere MQ V7.0, WebSphere MQ classes for JMS no longer have to poll a destination to check whether a suitable message has been sent to the destination or not. When a suitable message for an application arrives on the destination, WebSphere MQ server passes the message to the registered callback function. This reduces the inevitability for the JMS client to keep on polling for the messages. The advantage of using Asynchronous consume are listed below:

- Performance of JMS message listeners is improved, particularly when an application uses multiple message listeners in a session to monitor multiple destinations.
- Reduced usage of the CPU at both the JMS client and the WebSphere MQ server.
- Fewer requests over TCP/IP by the client to the server. Reduction in network traffic and I/O resource usage between client and server.
- Message throughput is increased, and the time taken to deliver a message to a message listener after it has arrived at a destination is reduced.

Similar enrichment are seen when Message Driven Beans (MBDs) are being used to retrieve messages asynchronously. Instead of the MDB’s polling for messages, WMQ passes the messages to be consumed by the MDB. In several situations there can be multiple MDB’s contending for consuming messages from the same destination. In such situations the race for message consumption results in higher CPU usage. With the enhancements for Asynchronous consumption with WebSphere MQ V7.0, multiple MDBs that are consuming messages from the same destination now experience reduced contention on the messages. This results in a higher performance for processing messages.

7.4 Conversation sharing sessions

Conversation sharing (Multiplexing) is an augmented feature that is integrated with WebSphere MQ V7.0. Conversation sharing allows a single TCP/IP socket to carry multiple connections or sessions provided the two ends of the connections belong to the same process. All Java Message Service (JMS) applications by default uses Multiplexing of sessions without any client code modifications.

In previous versions of WebSphere MQ, each instance of the JMS session created by the same parent JMS connection was using different socket connections to connect to the queue manager. No any two JMS sessions were sharing the same socket connection. JMS applications using multiple threads tend to create several JMS session objects from a single connection object for parallel processing of messages. This resulted in a new TCP/IP socket connection getting established for every single JMS session. The effect was that more Input/Output (I/O) resources were being used on both the Client and the Server.

With conversation sharing sessions integrated with WebSphere MQ V7.0, all the JMS sessions created from the same connection are multiplexed across a single TCP/IP socket. This implies that a JMS application creating multiple JMS sessions from a single connection object now uses only a single TCP socket. One consequence of multiplexing is that at both ends of the socket we now have a thread which is always receiving data on the socket. This allows the heartbeat down the socket from both the ends. In the event of a connection failure, because a single TCP/IP socket is used for multiple JMS sessions, the JMS client can immediately identify the connection breaking. The JMS application can be more responsive in identifying the failure all the times and not just when an MQGET is outstanding. Conversation sharing thereby reduces the dependency of KEEPALIVE on the TCP/IP.

Note: JMS Connection and JMS Session are JMS terminologies, for more information on these topics, refer JMS specifications.

KEEPALIVE: KEEPALIVE is a TCP/IP property that specifies for what duration the connection should be maintained if there are no activity on the socket.

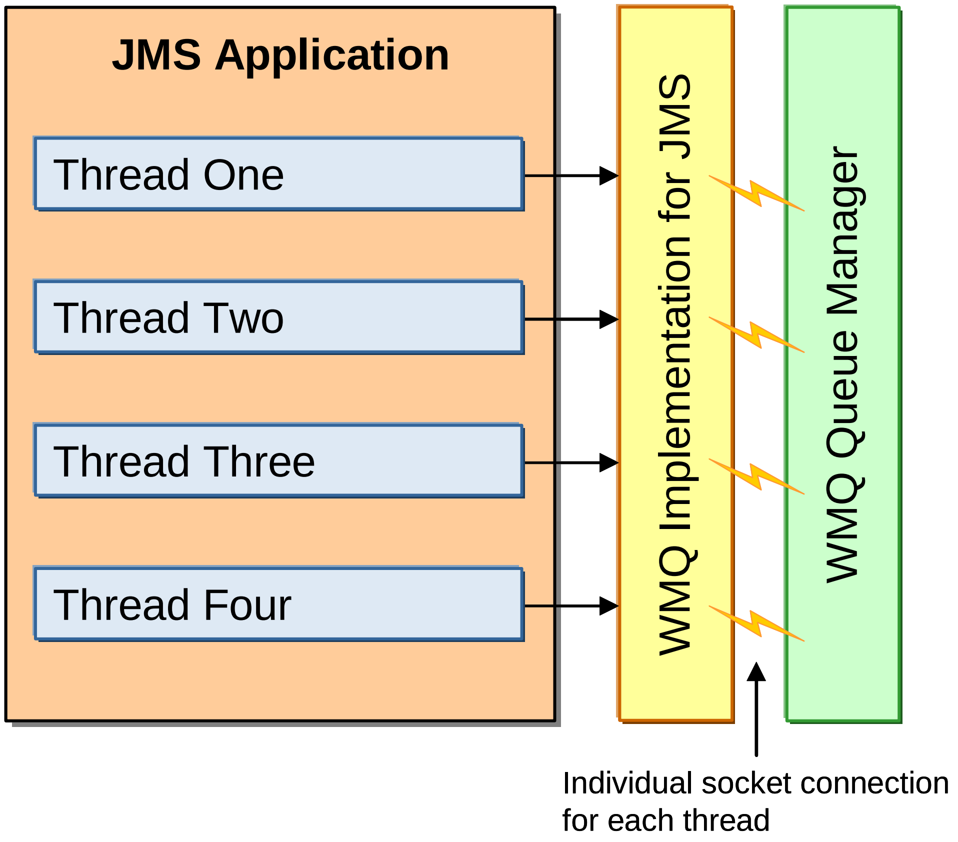


Figure 7-4 JMS multithreaded clients using WMQ V6.0

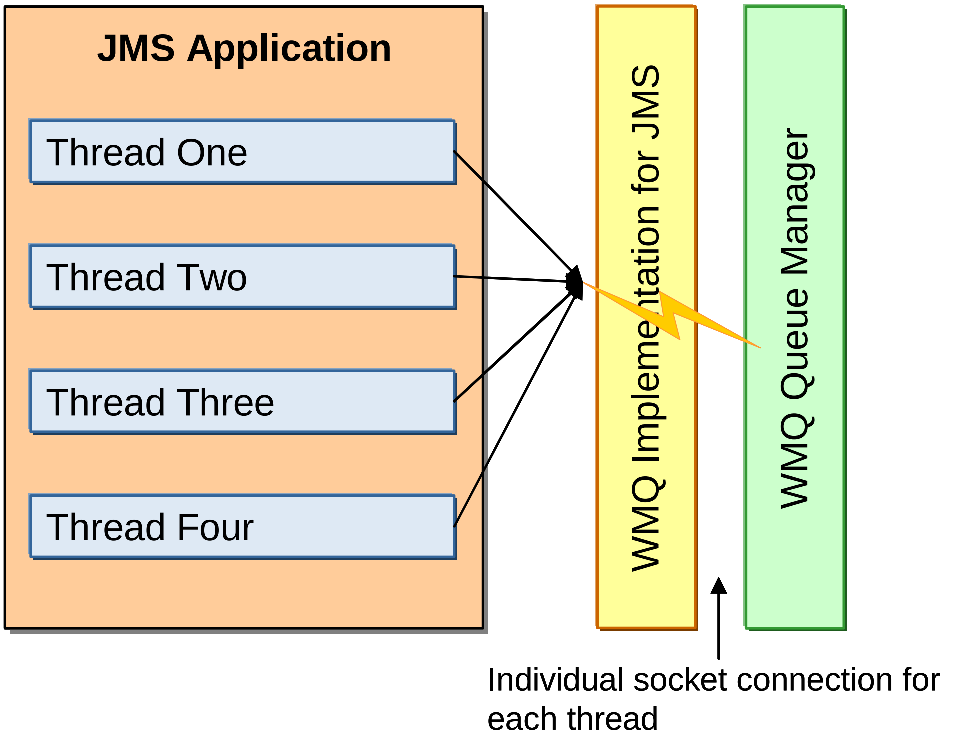


Figure 7-5 JMS multithreaded clients using WMQ V7.0

For more information on conversation sharing refer to the section 5.3, “Conversation sharing” on page 66.

7.5 Mapping of WebSphere MQ and JMS message

Java Message Service (JMS) client application use message selectors to filter for suitable messages from the destination. JMS client application receives only those messages containing a header field or a property matching the selector string specified by the application. In WebSphere MQ V6.0, the queue manager did not support message selection natively. The JMS client application had to browse the queue and perform the selection of messages itself. In situations, where there are thousands of messages on the destination, JMS client had to browse all the messages manually to identify suitable messages. This induced a lot of high CPU usage at both the client and the server side.

A JMS message consists of a set of header fields, a set of properties, and a body that contains the application data. As a minimum, a WebSphere MQ message consists of a message descriptor and the application data.

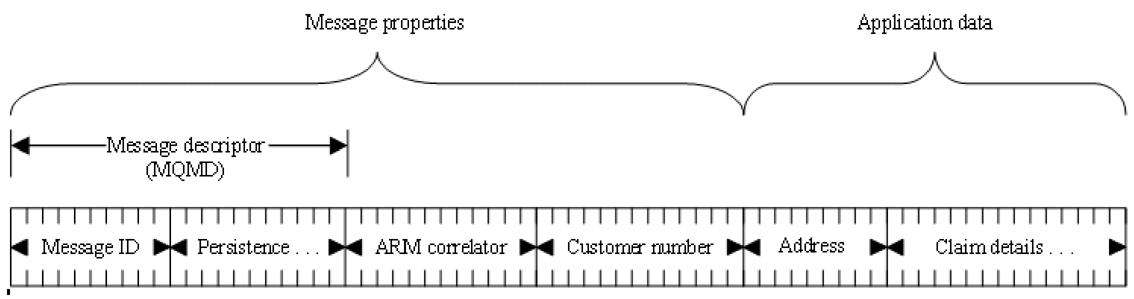


Figure 7-6 Representation of a message

WebSphere MQ V7.0 provides native MQ support for message selectors. This means the selection of messages is done by the queue manager itself. When a JMS application attempts to retrieve a message from the destination, the queue manager filters the message that match the selector and delivers back to the application. Message properties enable message selection on the Message Queue Interface (MQI).

One of the biggest problem customers have today when interoperating between JMS applications and MQ API application is the use of message properties. The JMS applications could not access the message properties in the MQ Message Descriptor (MQMD). The MQI application could not read the JMS header message properties. Message Properties in WebSphere MQ V7.0 allow customers to set and get a user property in a message, whether using JMS or using MQ API, with the same property name.

When a JMS application sends a JMS message, WebSphere MQ implementation for JMS maps the JMS message into a WebSphere MQ message. Some of the JMS header fields and properties are mapped into fields in the message descriptor, and some are mapped into fields in an additional WebSphere MQ header called an MQRFH2 header. When the JMS application receives a JMS message, WebSphere MQ implementation for JMS performs the reverse mapping.

An application that is using the MQI to receive messages from a JMS application must therefore be able to handle an MQRFH2 header. If the application cannot handle an MQRFH2 header, the TARGCLIENT property of the Destination object can be set to tell WebSphere MQ implementation for JMS not to include an MQRFH2 header in the WebSphere MQ messages. However, by excluding the MQRFH2 header, the information held in some of the JMS header fields and properties is lost.

Similarly, an application that is using the MQI to send messages to a JMS application must include an MQRFH2 header in each message. If an MQRFH2 header is not included, WebSphere MQ implementation for JMS can set only those JMS header fields and properties that can be derived from the fields in a message descriptor.

For more information about message properties refer to the section 6.2, “Message properties” on page 91.

7.6 Properties of WebSphere MQ classes for Java Message Service

All objects in WebSphere MQ classes for Java Message Service (JMS) have properties. Different properties apply to different object types. Different properties have different allowable values, and symbolic property values differ between the administration tool and program code.

WebSphere MQ classes for JMS provides facilities to set and query the properties of objects using the WebSphere MQ JMS administration tool, WebSphere MQ Explorer, or in an application. Many of the properties are relevant only to a specific subset of the object types.

This section discusses some of the new properties added with WebSphere MQ V7.0

Table 7-1 New properties and their corresponding Object types.

Property	**Short name**	**Description**	**CF/ XACF**	**QCF/ XAQCF**	**TCF/ XATCF**	**Q**	**T**
ASYNEXCEPTION	AEX	This property determines whether WebSphere MQ classes for JMS informs an ExceptionListener only when a connection is broken, or when any exception occurs asynchronously to a JMS API call. This applies to all Connections created from this ConnectionFactory that have an ExceptionListener registered.	Y	Y	Y		
MAPNAMESTYLE	MNST	Allows compatibility style to be used for MapMessage element names	Y	Y	Y		
PROVIDERVERSION	PVER	The version, release, modification level and fix pack of the queue manager to which the application intends to connect.	Y	Y	Y		

Property	Short name	Description	CF/ XACF	QCF/ XAQCF	TCF/ XATCF	Q	T
PUTASYNCALLOWED	PAA	Whether message producers are allowed to use asynchronous puts to send messages to this destination				Y	Y
READAHEADALLOWED	RAA®	Whether message consumers and queue browsers are allowed to use read ahead to get non-persistent messages from this destination into an internal buffer before receiving them				Y	Y
SENDCHECKCOUNT	SCC	The number of send calls to allow between checking for asynchronous put errors, within a single non-transacted JMS session	Y	Y	Y		
SHARECONVALLOWED	SCA	Whether a client connection can share its socket with other top-level JMS connections from the same process to the same queue manager, if the channel definitions match	Y	Y	Y		
SENDEXITINIT	SDXI	The user data that is passed to channel send exits when they are called	Y	Y	Y		
WILDCARDFORMAT	WCFM T	Which version of wildcard syntax is to be used	Y		Y		Y

Besides the above mentioned properties, WebSphere MQ V7.0 introduces new attributes, which can be used with the JMS administrative objects. For a complete reference of the properties and their detailed description refer the WebSphere MQ V7.0 Infocenter to the section *Using Java* → *WebSphere MQ classes for JMS* → *Properties of objects*, available at:

https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.uj.doc/jm10910_.htm

8

Administration enhancements

This chapter discusses the enhancements and changes of WebSphere MQ V7.0 from the administration point of view. The administration from WebSphere MQ Explorer and command-line interface is covered.

The following topics are discussed.

- “WebSphere MQ Explorer”
- “Working with new properties and parameters”
- “Java and JMS related administration enhancements”
- “The control commands”
- “Journal time stamps on i5/OS”

The Publish/Subscribe administration is not covered in this chapter due to complexity of Publish/Subscribe. For these administration tasks refer to Chapter 9., “Publish/Subscribe management” on page 193.

8.1 WebSphere MQ Explorer

The Eclipse-based graphical tooling, MQ Explorer, introduced in the previous release is further updated in WebSphere MQ V7.0. MQ Explorer enables remote configuration of WebSphere MQ from Linux x86 and Windows machines, does not require a local server or client and can be installed on machines without charge.

This section describes what is new in MQ Explorer. The main enhancements are related to:

- General GUI enhancements: new MQ Explorer has a Welcome page to help new or less experience users and it is now possible to export or import MQ Explorer settings.
- Browsing messages: the window have been enhanced to display message properties. Message properties are new extension to message structure in WebSphere MQ V7.0.
- Mapping between MQ objects and JMS objects: new JMS administered objects can be created based on your existing MQ objects or new MQ objects based on your existing JMS object.
- Remote queue managers administration: connection details to remote queue manager have been enhanced to enable define specific connection for each queue manager instead of for whole MQ Explorer. Now it is also possible to check and define prerequisites for remote administration by one click from MQ Explorer.
- Security: improved capabilities to secure administration connection to your queue managers.
- Queue manager sets: queue managers can be grouped into the sets to issue administration task on set instead of on queue managers one by one.
- SupportPac MS0Q integration: the functionality of this SupportPac is now integrated into MQ Explorer to manage topics for V6 queue managers.

Several enhancements are fully compatible with V6 queue managers and MQ administrators can profit from these MQ Explorer V7.0 enhancements.

8.1.1 General GUI enhancements

Tip: These enhancements can be also used for V6 queue managers.

There are two general enhancements to MQ Explorer GUI. Welcome page brings better start point of using MQ Explorer to new or less experience users. Export and import of MQ Explorer settings are useful for product reinstallation or transfer settings to another instance of MQ Explorer (for example multiple installation).

Welcome page

This page automatically displayed when you launch MQ Explorer for the first time.

Tip: To display the Welcome page in any time later select **Help** → **Welcome** from the main menu.

The figure below shows the new MQ Explorer Welcome page:

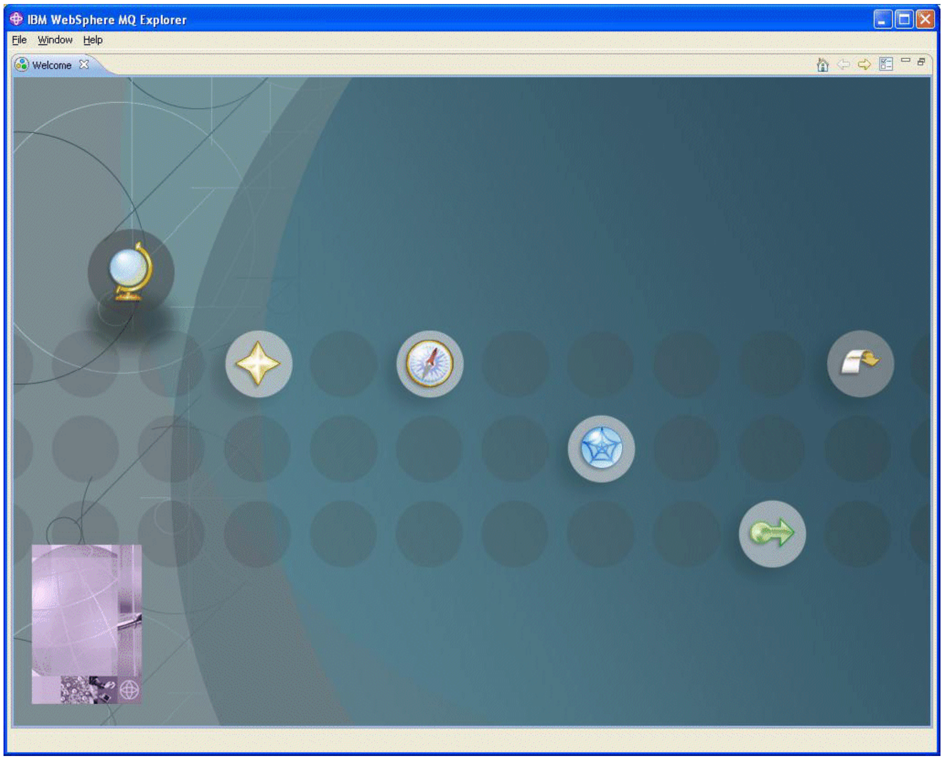


Figure 8-1 MQ Explorer Welcome page

There are six icons on the Welcome page and they are (from the left):

- Overview: this view contains links to the basic Eclipse information and MQ Explorer product tour and help.
- What is new: contains information about new features in Eclipse and Java development tools.

Author Comment: No information on the screen at the time of writing...

- First Steps: this view contains links to the WebSphere MQ default configuration setup and to the sample WebSphere MQ applications.
- Web Resources: this view contains links to the WebSphere MQ product family web pages and to the WebSphere MQ product support web pages.
- Migrate

Author Comment: This view contains no links at the time of writing...

- Workbench: click to this icon to proceed to the main MQ Explorer views. It closes Welcome page and if you want to open it again select **Help** → **Welcome** from the main menu.

Export and Import settings

Settings can be exported or imported from or to MQ Explorer. This can be useful for backup purposes, for example product reinstallation, or transfer settings to another instance of MQ Explorer, for example multiple installation.

The following types of settings can be exported or imported:

- MQ Explorer preferences
- Filters and Column schemes
- Connection Information (for remote queue managers)
- Queue manager Sets information (memberships, set definitions and filters)

To access the export and import capabilities for MQ Explorer right click on **IBM WebSphere MQ** in the Navigator view, as illustrated in the figure below:

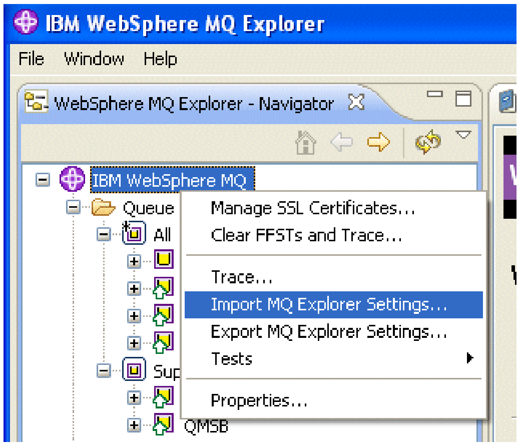
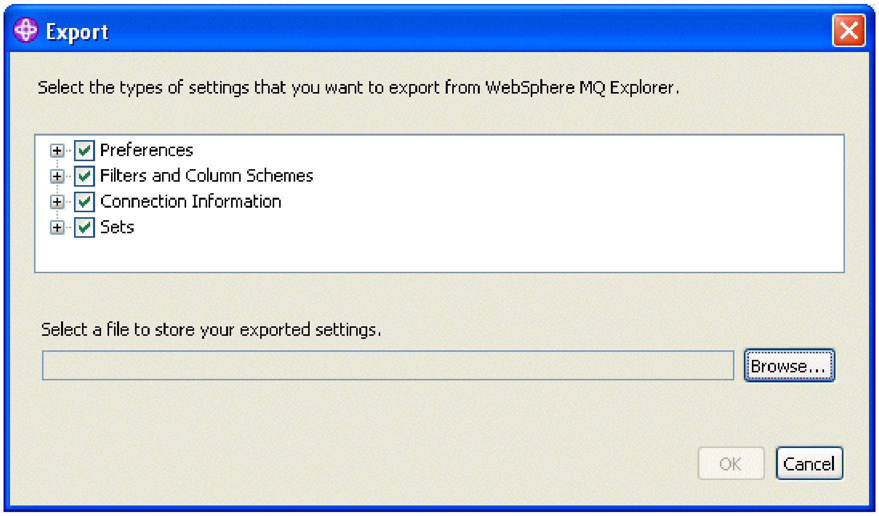

IBM WebSphere MQ Explorer
File Window Help
WebSphere MQ Explorer - Navigator
IBM WebSphere MQ
Queue
All
Sup
QMSB
Manage SSL Certificates...
Clear FFSTs and Trace...
Trace...
Import MQ Explorer Settings...
Export MQ Explorer Settings...
Tests
Properties...

Figure 8-2 Export and import settings in MQ Explorer V7.0

Exporting settings

To export settings click to **Export MQ Explorer Settings...** and then select which settings you want to export and the location in which the file needs to be stored. See the export dialog figure below:



Select the types of settings that you want to export from WebSphere MQ Explorer.

- [✓] Preferences
- [✓] Filters and Column Schemes
- [✓] Connection Information
- [✓] Sets

Select a file to store your exported settings.

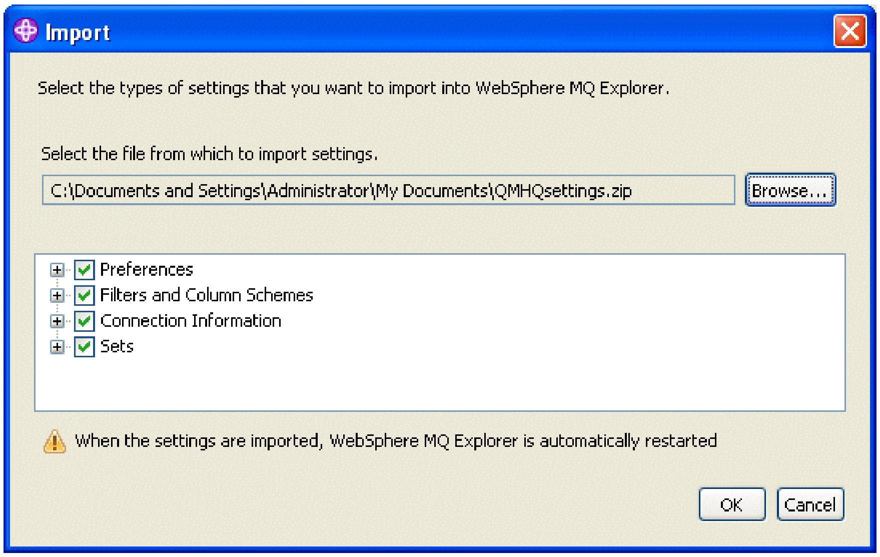
Figure 8-3 Export settings dialog

A ZIP file in XML format is created that contains the exported settings.

Note: When exporting manually created queue manager set (the set that was created from queue manager names) a list of the queue manager names and their QMIDs is exported.

Importing settings

To import settings click to **Import MQ Explorer Settings...** and then select the ZIP file for import and settings that you want to import. See the import dialog in the figure below:



Select the types of settings that you want to import into WebSphere MQ Explorer.

Select the file from which to import settings.

C:\Documents and Settings\Administrator\My Documents\QMHQsettings.zip

- ✓ Preferences
- ✓ Filters and Column Schemes
- ✓ Connection Information
- ✓ Sets

When the settings are imported, WebSphere MQ Explorer is automatically restarted

Figure 8-4 Import settings dialog

Note: For detailed information on how import works if your ZIP file contains manually created queue manager sets and filters, refer to the topic *Importing and exporting queue manager sets in WebSphere MQ Explorer* in the WebSphere MQ V7.0 Information Center at:

https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_sets_importexport.htm

Importing schemes and filters from MQ Explorer V6

Schemes and filters can be imported from the previous version of MQ Explorer. The schemes can be imported for queues, channels, listeners and also schemes for status tables in the Status dialogs (for example Queue Status dialog and Topic Status dialog).

Note: For detailed information about V6 import restrictions, refer to the topic *Exporting and importing settings in WebSphere MQ Explorer* in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_importexport.htm`

8.1.2 Browsing messages

WebSphere MQ Explorer V7.0 can display message properties. Message properties are new extension to message structure. Also, two MQ Explorer preferences related to message browsing have been added.

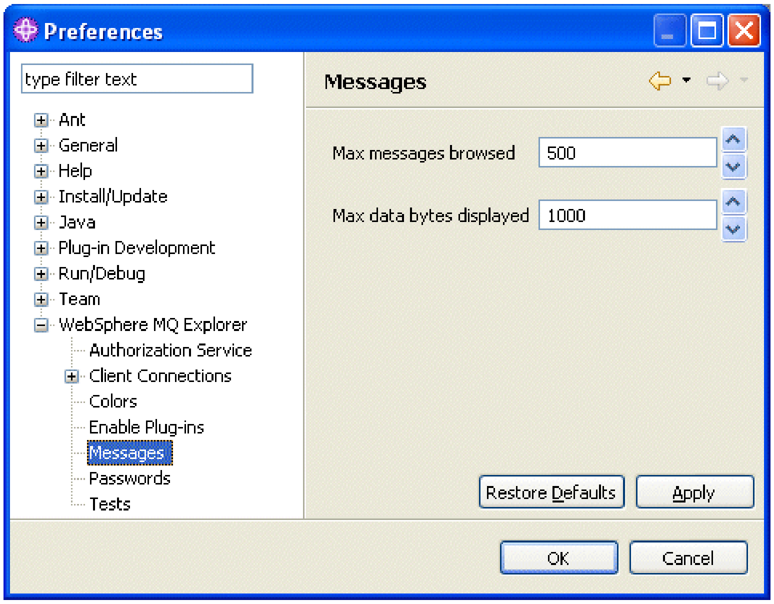
New MQ Explorer preferences

Tip: This enhancement can be also used for V6 queue managers.

Now you can specify how many message can be browsed and what size of the message content is displayed:

- Max messages browsed: maximum number of messages displayed in the Message browser window; default value is 500.
- Max data bytes displayed: the number of bytes (from the beginning of the message) which is displayed; default value is 1000.

To set these preferences go to the main menu and then select **Window** → **Preferences**. In the left pane of the window displayed, select **WebSphere MQ Explorer** → **Messages**. See the figure below:

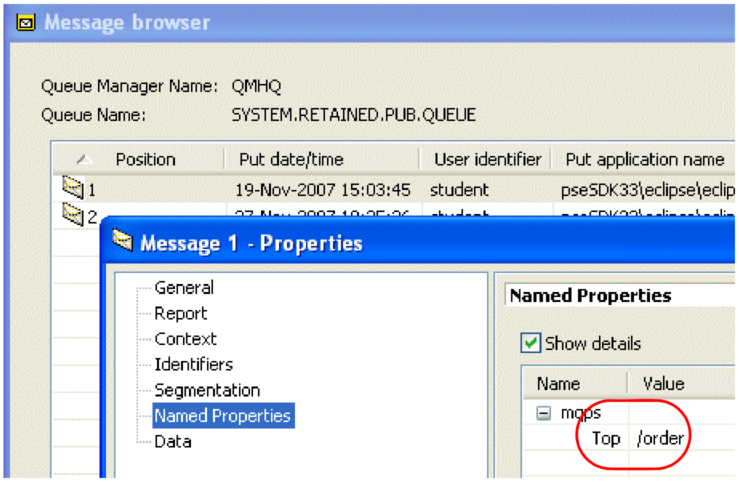


Max messages browsed: 500
Max data bytes displayed: 1000

Figure 8-5 Message browsing preferences

Browsing Message properties

The new option **Named Properties** has been added to message browsing window, as illustrated in the figure below:



Queue Manager Name: QMHQ
Queue Name: SYSTEM.RETAINED.PUB.QUEUE

Position	Put date/time	User identifier	Put application name
1	19-Nov-2007 15:03:45	student	pseSDK33\eclipse\eclip
2			

Name	Value
mqps	
Top	/order

Figure 8-6 Browsing Message properties

This example shows Publish/Subscribe message. See property **Top** (topic) with value /order under the **Named Properties** option.

8.1.3 Mapping between MQ objects and JMS objects

Tip: These enhancements can be also used for V6 queue managers.

These new MQ Explorer functions can be used to simplify your administration tasks when creating MQ object as queues or topics.

Creating a MQ object from a JMS object

New MQ queues and topics can be created based on your existing JMS queues and topics. The values of relevant properties of the JMS object are copied to the new MQ object.

Important: If you make a change to one of the objects after the creation has been done, the changes are not reflected in the other object.

To create new MQ object from an existing JMS object:

1. Expand your **JMS Administered Objects** tree in the Navigator view and select source JMS object, queue or topic, from the **Destinations** folder.
2. Right-click the object and select **Create MQ Queue...** or **Create MQ Topic...** . The wizard opens as appropriate.
3. Work through the wizard to define the new MQ object, then click **Finish**.

   Note: If the JMS object used to create a MQ object specifies a queue manager name in its properties, then the MQ object can only be created on the queue manager with the same name.

4. The new MQ object is created and displayed under the appropriate queue manager in MQ Explorer.

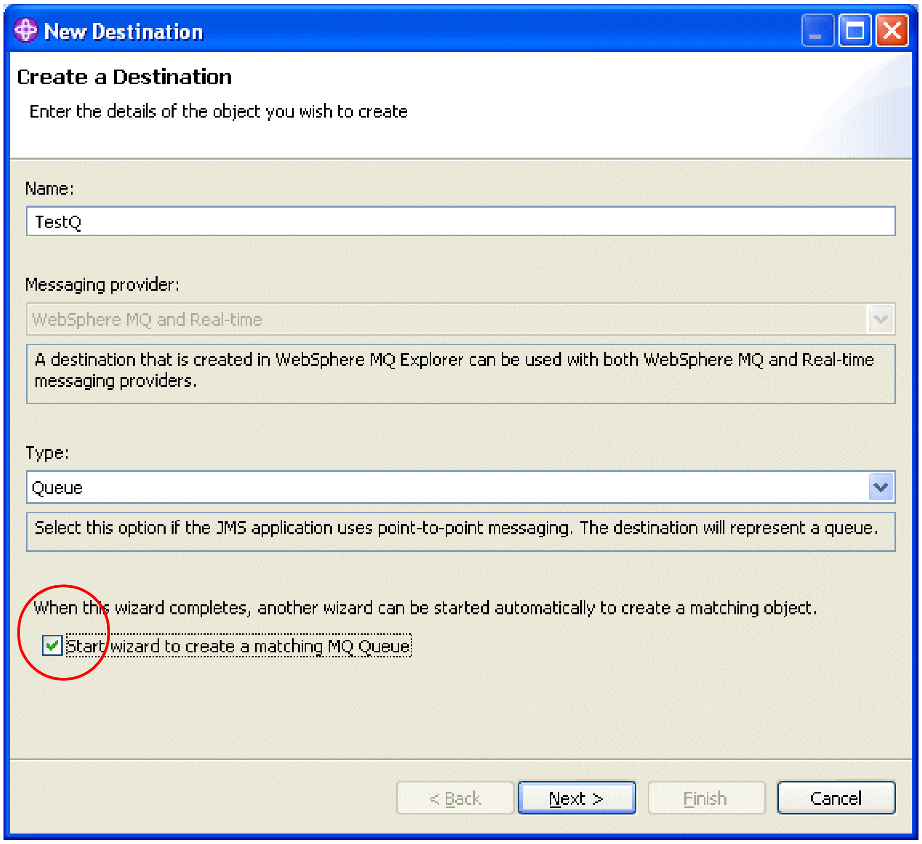
Creating a JMS object and an MQ object simultaneously

When creating a new JMS object (queue or topic) in MQ Explorer and after it has completed successfully, an appropriate MQ object can be created immediately.

To create a new JMS object and MQ object simultaneously:

1. Create the new JMS object as usual.

2. Select checkbox **Start wizard to create a matching MQ** object, as illustrated in the figure below (for the queue):



Create a Destination

Enter the details of the object you wish to create

Name: TestQ

Messaging provider: WebSphere MQ and Real-time

A destination that is created in WebSphere MQ Explorer can be used with both WebSphere MQ and Real-time messaging providers.

Type: Queue

Select this option if the JMS application uses point-to-point messaging. The destination will represent a queue.

When this wizard completes, another wizard can be started automatically to create a matching object.

✓ Start wizard to create a matching MQ Queue

Figure 8-7 Create an MQ object simultaneously

3. The creating MQ object wizard launches immediately after creating the new JMS object has finished.

Note: The detail on creating a MQ object from a JMS object is covered in the following topics. Refer to the WebSphere MQ V7.0 Information Center:

- *Creating a WebSphere MQ object from a JMS object*

  `https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/j_mqfromjms_creating.htm`

- *Creating a JMS object and an MQ object simultaneously*

  `https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/j_creating_jms_and_mq.htm`

Creating a JMS object from a MQ object

New JMS queues and topics can be created based on your existing MQ queues and topics. The values of relevant properties of the MQ object are copied to the new JMS object.

Important: If you make a change to one of the objects after the creation has been done, the changes are not reflected in the other object.

To create new JMS object from an existing MQ object:

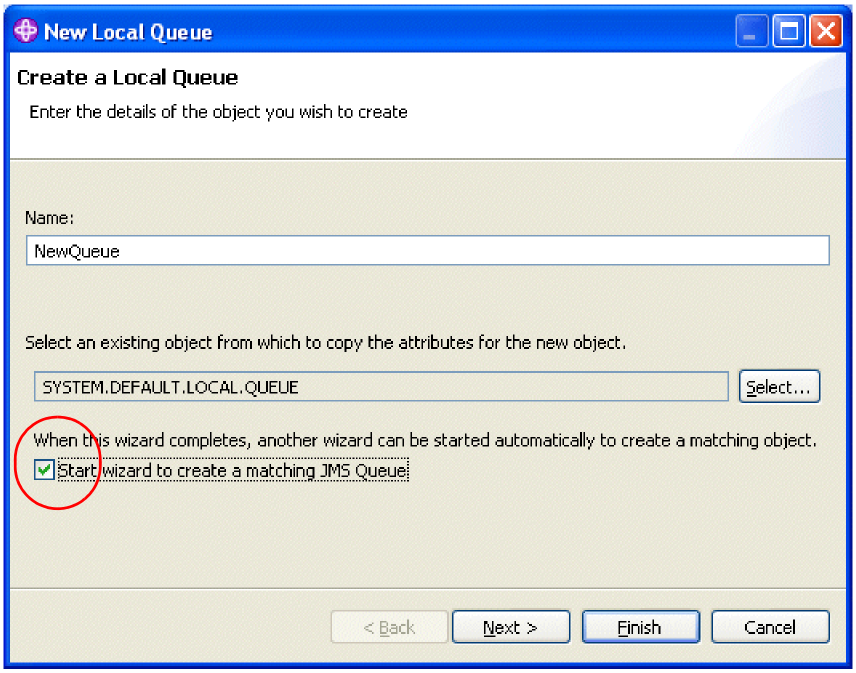
1. Expand the queue manager objects tree in the Navigator view and select source MQ object, queue or topic, in the Content view.
2. Right-click the object and select **Create JMS Queue...** or **Create JMS Topic...** . The wizard opens as appropriate.
3. Work through the wizard to define the new JMS object, then click **Finish**.
4. The new JMS object is created and displayed under the Destination folder in the **JMS Administered Objects** tree.

Creating an MQ object and a JMS object simultaneously

When creating a new QM object (queue or topic) in MQ Explorer, and after it has completed successfully an appropriate JMS object can be created immediately.

To create a new MQ object and JMS object simultaneously:

1. Create your new MQ object as usual.
2. Select checkbox **Start wizard to create a matching JMS object**, as illustrated at the figure below (for the queue):


New Local Queue

Create a Local Queue

Enter the details of the object you wish to create

Name:

NewQueue

Select an existing object from which to copy the attributes for the new object.

SYSTEM.DEFAULT.LOCAL.QUEUE

When this wizard completes, another wizard can be started automatically to create a matching object.

[✓] Start wizard to create a matching JMS Queue

Figure 8-8 Create a JMS object simultaneously

3. The creating JMS object wizard launches immediately after the creating the new MQ object has finished.

Note: The detail on creating a JMS object from a MQ object are covered in the two following topics. Refer to the WebSphere MQ V7.0 Information Center:

- *Creating a JMS object from a WebSphere MQ object*

  `https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/j_jmsfrommq_creating.htm`

- *Creating an MQ object and a JMS object simultaneously*

  `https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_creating_mq_and_jms.htm`

8.1.4 Remote queue managers administration

The remote administration capabilities have been significantly enhanced to administer remote queue managers in an easy and secure way.

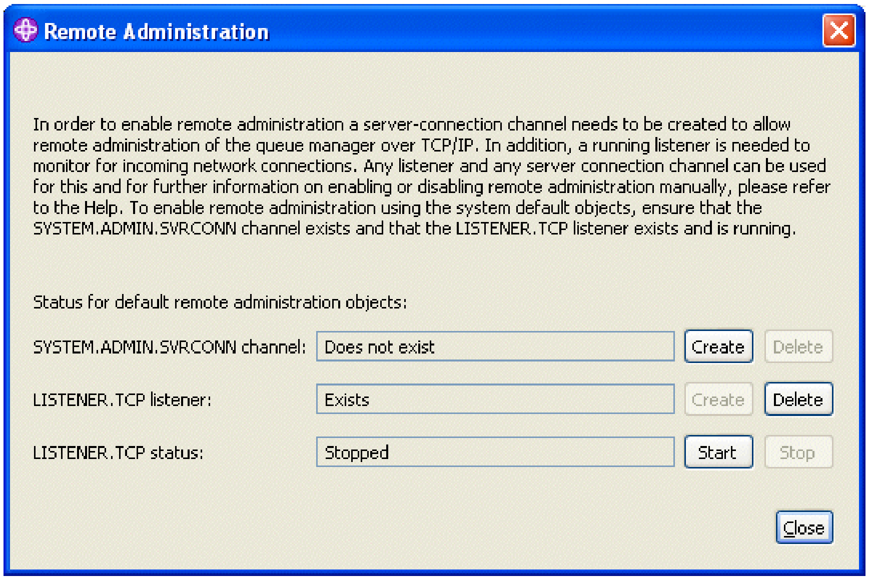
The administration enhancements covered in this section are about working with remote queue managers (how to add / show /hide a queue manager and how to check if all prerequisites for remote administration are set properly). There are also security related enhancements for remote queue managers administration which are covered in the next section, 8.1.5, “Security”.

Remote Administration dialog

The following objects /settings are needed for remote queue manager administration:

- The MQ command server has to be running
- The MQ listener has to be running
- The server-connection channel SYSTEM.ADMIN.SVRCONN has to exist
- The model queue SYSTEM.MQEXPLORER.REPLY.MODEL has to exist

The new Remote Administration dialog is now available for local queue managers to check and create these prerequisites. To use it, select appropriate local queue manager, right-click it and then select **Remote Administration...** option. The Remote Administration dialog is displayed, as described on the figure below:



In order to enable remote administration a server-connection channel needs to be created to allow remote administration of the queue manager over TCP/IP. In addition, a running listener is needed to monitor for incoming network connections. Any listener and any server connection channel can be used for this and for further information on enabling or disabling remote administration manually, please refer to the Help. To enable remote administration using the system default objects, ensure that the SYSTEM.ADMIN.SVRCONN channel exists and that the LISTENER.TCP listener exists and is running.

Status for default remote administration objects:

SYSTEM.ADMIN.SVRCONN channel: Does not exist

LISTENER.TCP listener: Exists

LISTENER.TCP status: Stopped

Figure 8-9 The Remote Administration dialog

Notes: The **Remote Administration...** option is only available if a queue manager is running.

The Remote Administration dialog doesn't cover MQ command server and SYSTEM.MQEXPLORER.REPLY.MODEL queue because these object are created by default when a queue manager is created.

This dialog can be used to maintain SYSTEM.ADMIN.SVRCONN channel and TCP/IP listener by click to appropriate radio buttons. All action are single one-click operations except **Create** for TCP/IP listener, which requires input information about TCP/IP port number. The wizard also warning you if you type the port number which is already used by another queue manager.

Note: For detailed information, refer to the topic *Enabling remote administration of queue managers* in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_gui_remote_enabling.htm`

Showing and hiding a remote queue manager

Tip: This enhancement can be also used for V6 queue managers.

There are some changes about working with remote queue managers:

- You can add a queue manager for remote administration directly from context menu with the new add option instead of opening whole Show/Hide Queue Managers window.

  To add a remote queue manager right-click on **Queue Managers** folder in the Navigator view, then select **Add Remote Queue Manager...** option.

- You can hide your remote queue manager also directly from context menu with the new hide option instead of opening whole Show/Hide Queue Managers window.

  To hide your remote queue manager right-click on queue manager name folder in the Navigator view, then select **Hide** option.

You can still add or hide your queue managers in the old fashion style from the Show/Hide Queue Managers window.

8.1.5 Security

Tip: These enhancements can be also used for V6 queue managers.

This section describes security related enhancements for WebSphere MQ Explorer V7.0 which enable to use more secure administration connections to queue managers.

The new enhancements enable to secure administration connection to a queue manager with user ID and password and to specify different security exits per each queue manager.

Note: There was possible to set up only one common security exit (for all managed queue managers) in the previous versions of WebSphere MQ Explorer.

The new security related parameters can be set at the following two places in the MQ Explorer:

- In MQ Explorer preferences, which can be used as default for any specific connections.
- For each remote queue manager connection.

The details about security exits for queue manager client connection are described in the section 5.10, “Security exit details in WebSphere MQ Explorer” on page 84.

This section only describes where the security exits can be set up in MQ Explorer for remote queue managers administration purpose.

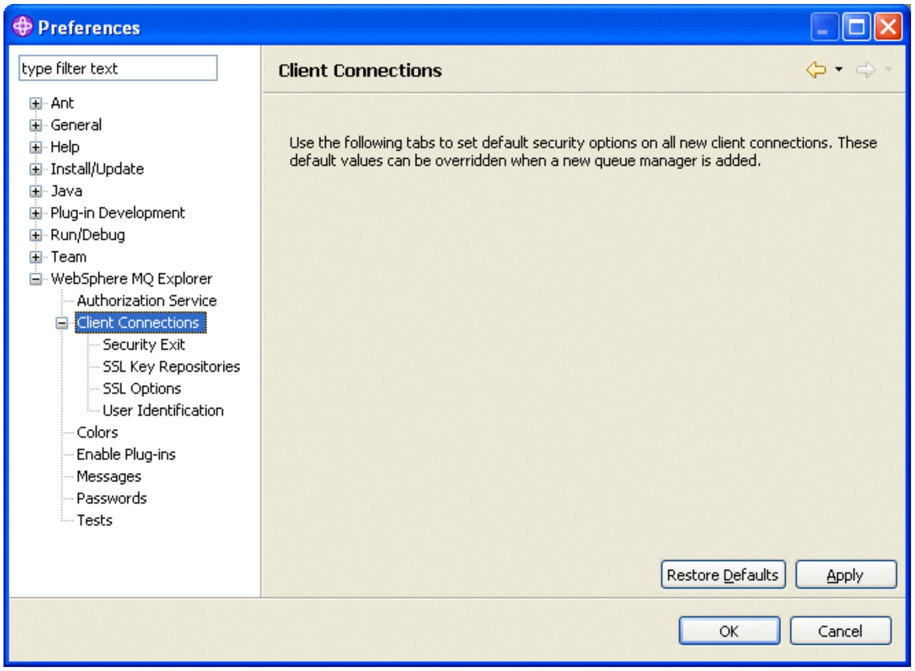
Default security preferences

The new options in MQ Explorer preferences have been add to support new security features, **Client Connections** and **Passwords**.

Client connections

The default security exit and default user ID / password parameters for all the administered queue managers can be set and changed here. The parameters can be overridden when define a new security options during add a new remote queue manager.

To set default preferences for client connections go to the main menu and then select **Window** → **Preferences**, the Preferences window is displayed. In the left pane select **WebSphere MQ Explorer** → **Client Connections** to see the following screen:



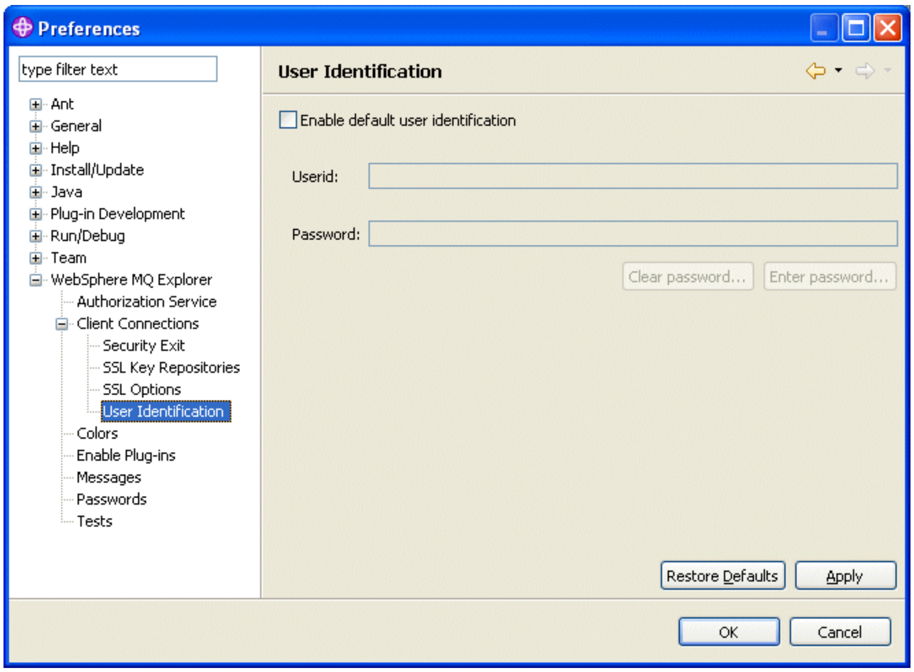
Use the following tabs to set default security options on all new client connections. These default values can be overridden when a new queue manager is added.

Figure 8-10 Default Client Connections settings

There are four next options under the Client Connections option:

- Security Exit: to set up default security exit name, path and exit data (parameters).
- SSL Key Repositories: to set up default SSL trusted certificate store and personal certificate store.
- SSL Options: to set up default cipher specification, SSL reset count and peer name.
- User Identification: to set up default user ID and password for client connections.

User Identification is another new capability in MQ Explorer v7.0. Now is it possible to secure administration connection to a queue manager with used ID and password. This screen is illustrated at the figure below:



User Identification

☐ Enable default user identification

Userid:

Password:

Figure 8-11 Default User Identification settings

The only Userid can be enter and password dialog is displayed each time during the connection process. Or the password can be enter and save. The password is stored in password store in the secure way. In this case, the parameters for password store have to be set. This is described in the next section Passwords below.

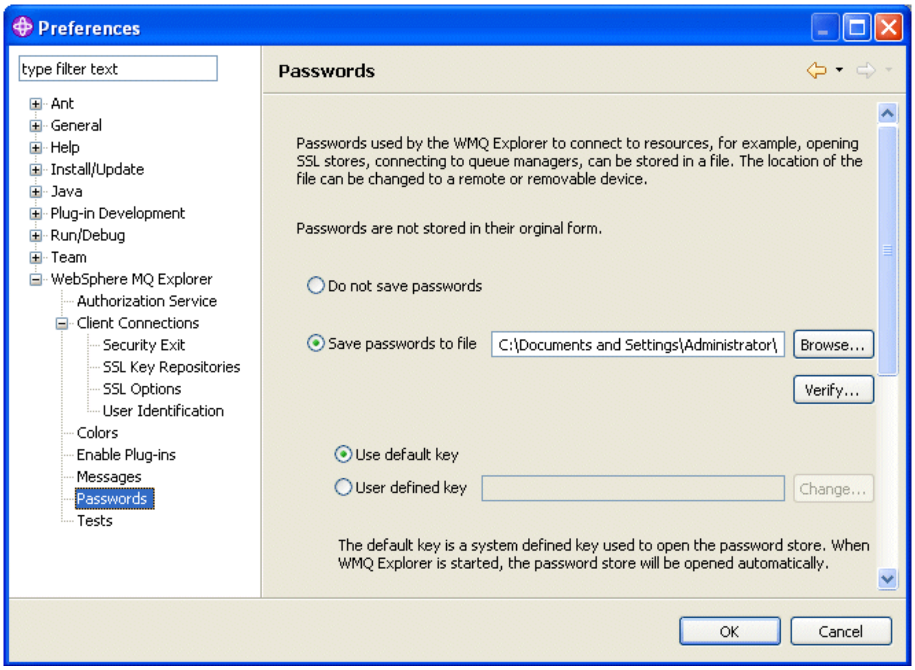
Note: For detailed information, refer to the topic *Default security preferences* in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_default_sec_pref.htm`

Passwords

Passwords used by the MQ Explorer to connect to resources (for example opening SSL stores or connecting to queue managers), can be stored in a file.

To set default preferences for passwords go to the main menu and then select **Window** → **Preferences**, the Preferences window is displayed. In the left pane select **WebSphere MQ Explorer** → **Passwords** to see the following screen:



Passwords used by the WMQ Explorer to connect to resources, for example, opening SSL stores, connecting to queue managers, can be stored in a file. The location of the file can be changed to a remote or removable device.

Passwords are not stored in their orginal form.

Do not save passwords [✓ Save passwords to file] C:\Documents and Settings\Administrator\

[✓ Use default key] User defined key

The default key is a system defined key used to open the password store. When WMQ Explorer is started, the password store will be opened automatically.

Figure 8-12 Default Password settings

To enable or disable saving password feature, choose the file which store passwords and key for password store.

To check if the password store file exists and have appropriate permissions click on the radio button **Verify...** .

With option **Use default key** the password store opens automatically when is needed. The option **User defined key** means the password store has to be accessed with specified password. In this case the password dialog is always open when the password store needs to be open for the first time after MQ Explorer has launched.

Note: For detailed information, refer to the topic *Passwords preferences* in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_password_pref.htm`

Security settings for remote queue manager connection

The **Add Remote Queue Manager...** wizard has four new screens (dialogs) to define specific security connection information for each queue manager.

The first two screens are the same as for the previous version of MQ Explorer which allow to define basic connection parameters (queue manager name, TCP/IP address and port, refresh interval). Then the wizard navigates you through the following screens:

- Specify security exit details: to set up default security exit name, path and exit data (parameters).
- Specify user identification details: to set up default user ID and password for client connections.
- Specify SSL certificate key repository details: to set up default SSL trusted certificate store and personal certificate store.
- Specify SSL option details: to set up default cipher specification, SSL reset count and peer name.

Note: For detailed information, refer to the section *Creating a new security-enabled connection* in the topic *Showing a remote queue manager* in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_qmanager_showremote.htm`

8.1.6 Queue manager sets

Tip: These enhancements can be also used for V6 queue managers.

MQ Explorer now supports the grouping of queue managers. Group of queue manager is called a Set. Actions may be performed on a set of queue managers instead of on queue managers one by one.

This enables to subdivide your queue managers to separate sets, for example based on the environment (`Test`, `Production...`), the version of WebSphere MQ (`Version6`, `Version7`), departments of your company or on the operating system.

The following actions can be performed on a set:

- Show/Hide all queue managers

  Only displayed if there is at least one hidden/visible queue manager in the set.

- Connect/Disconnect all queue managers

  Only displayed if there is at least one disconnected/connected queue manager in the set.
- Start/Stop all local queue managers

  Only displayed if there is at least one stopped/started local queue manager in the set.
- Run Default/Custom tests
- Edit connection details
- Add new local or remote queue manager

  The new queue manager become automatically member of this set.

Grouping can be done manually (you simply select queue managers which you want to add to set) or automatically, by filtering on:

- Command level
- Platform
- Any other queue manager attribute by custom made filter

Queue managers may be members of none, one or many sets. Sets may not contain other sets. There is default set called **All** which cannot be edited or deleted.

Working with the queue manager sets

At first, displaying sets in MQ Explorer has to be enable. Although the queue manager sets still exist when the sets are hidden, they cannot be managed.

Enable displaying sets in MQ Explorer

To display sets perform:

1. Right-click the **Queue Managers** folder in the Navigator view.
2. Select **Sets** → **Show Sets**.
3. The command displays a default set called **All** which contains all the queue managers.

To hide all the sets perform:

1. Right-click the **Queue Managers** folder in the Navigator view.
2. Select **Sets** → **Hide Sets**.
3. The command removes all the defined sets, including the **All** set, from the Navigator view but all defined sets are not deleted and still exist.

The option **Sets** in the **Queue Managers** folder context menu contains all action for set management, as illustrated on the figure below:

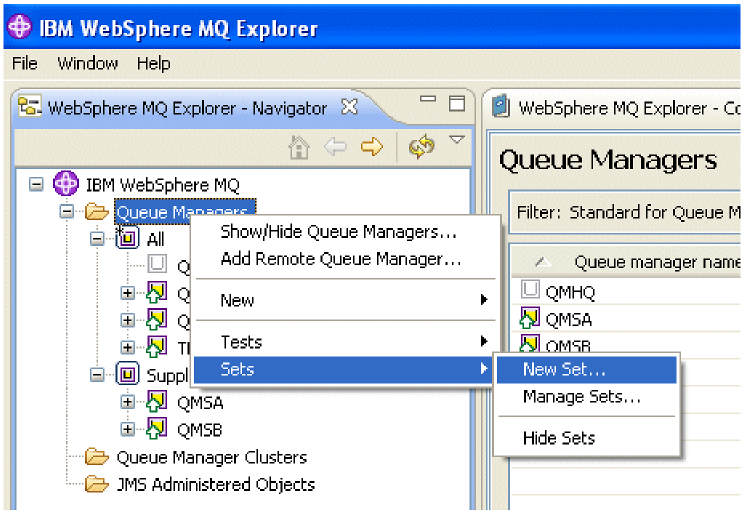


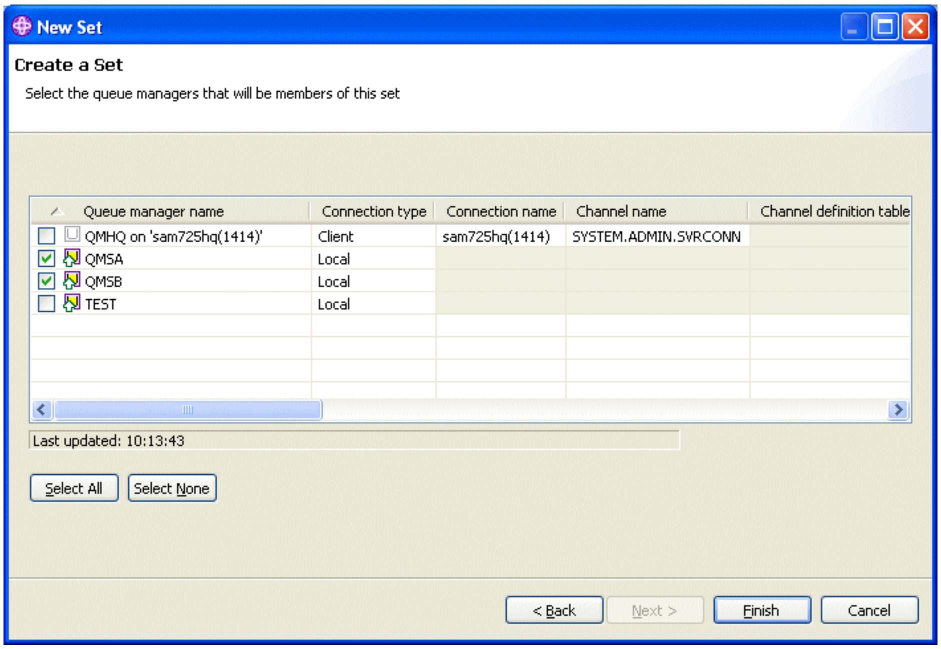
Figure 8-13 Context menu Sets

From this menu create a new set, manage existing sets or Show/Hide all sets.

Creating a new set

To add a new set perform:

1. Select option New Set... from the Sets context menu.
2. Work through the wizard to define the new set.
3. Enter name for new set and select its type in the first screen.
4. In the next screen, select queue managers (for manual set) or filters (for automatic set), as illustrated on the two figures below:



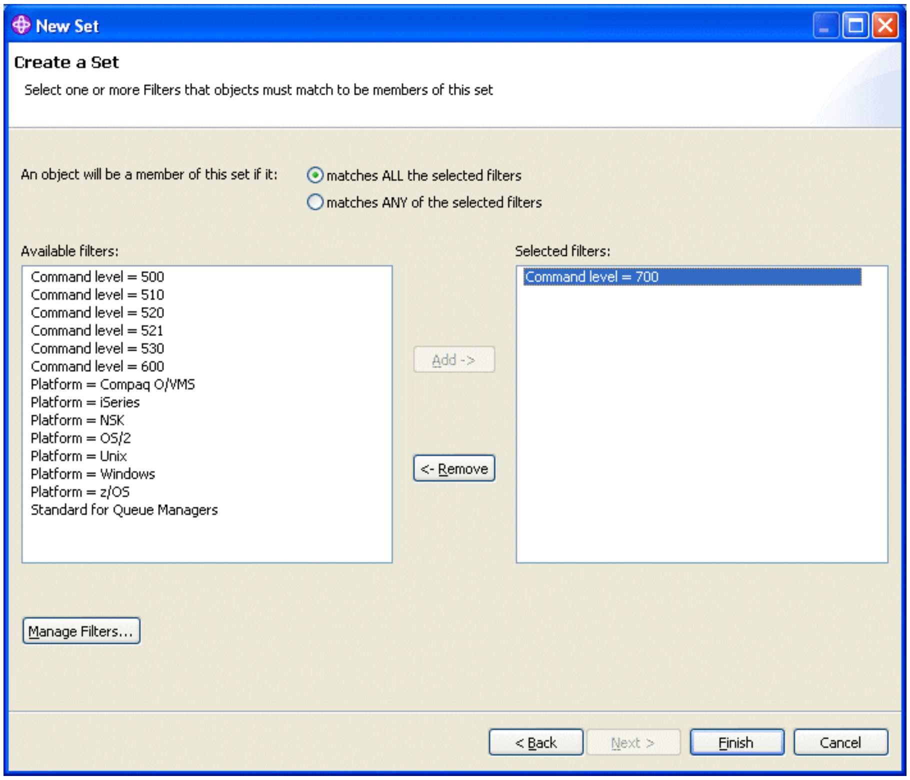
Create a Set

Select the queue managers that will be members of this set

	Queue manager name	Connection type	Connection name	Channel name	Channel definition table
☐	QMHQ on 'sam725hq(1414)'	Client	sam725hq(1414)	SYSTEM.ADMIN.SVRCONN	
✓	QMSA	Local			
✓	QMSB	Local			
☐	TEST	Local			

Last updated: 10:13:43

Figure 8-14 Queue manager selection for manual set



Create a Set

Select one or more Filters that objects must match to be members of this set

An object will be a member of this set if it: [✓ matches ALL the selected filters] matches ANY of the selected filters

Available filters:

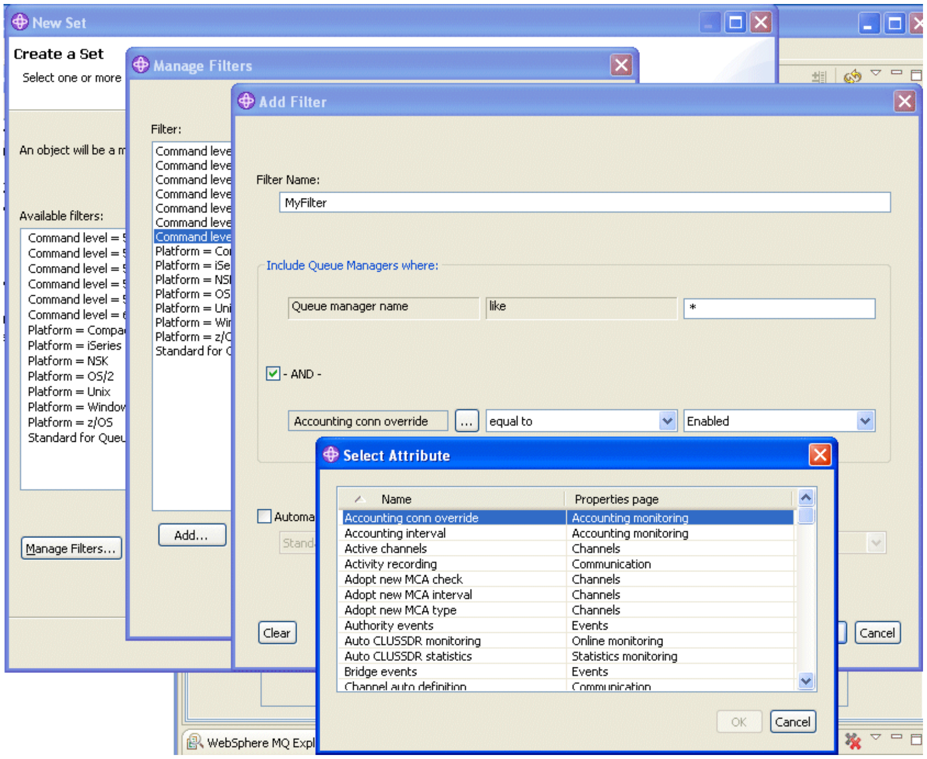
Command level = 500
Command level = 510
Command level = 520
Command level = 521
Command level = 530
Command level = 600
Platform = Compaq O/VMS
Platform = iSeries
Platform = NSK
Platform = OS/2
Platform = Unix
Platform = Windows
Platform = z/OS
Standard for Queue Managers

Selected filters:

Command level = 700

Figure 8-15 Filters selection for automatic set

5. In case of manual set then click **Finish**.
6. In case of automatic set it is possible to define custom filter. Click to **Manage Filters...** → **Add...** then define your rules. To display window with queue manager attributes click to **...** (three dots) radio button, see the figure below:



Filter Name: MyFilter

Include Queue Managers where:

Queue manager name	like	*

[✓] - AND -

Accounting conn override	...	equal to	Enabled

Select Attribute

Name	Properties page
Accounting conn override	Accounting monitoring
Accounting interval	Accounting monitoring
Active channels	Channels
Activity recording	Communication
Adopt new MCA check	Channels
Adopt new MCA interval	Channels
Adopt new MCA type	Channels
Authority events	Events
Auto CLUSSDR monitoring	Online monitoring
Auto CLUSSDR statistics	Statistics monitoring
Bridge events	Events
Channel auto definition	Communication

OK Cancel

Figure 8-16 Custom filter definition

7. Make your filter then go back to the New Set wizard window and click **Finish**.

Using the sets

The defined sets can be used for administration tasks. Right-click the queue manager set name in the Navigator view and then select desired action, as illustrated on the figure below:

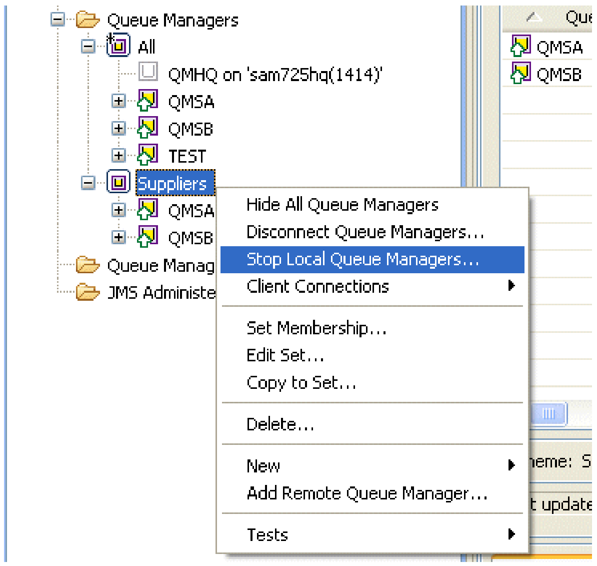


Figure 8-17 Queue manager set actions

Note: For all information about queue managers sets, refer to the topic *Creating and configuring a queue manager set* in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.explorer.doc/e_setoverview.htm`

8.1.7 SupportPac MS0Q integration

Tip: This enhancement is only related to V6 queue managers.

The functionality of SupportPac MS0Q: WebSphere MQ Explorer Publish/Subscribe plug-in is now integrated into MQ Explorer V7.0.

This functionality extends the MQ Explorer to provide a topics based view of the Publish/Subscribe broker at V6 queue manager. The topics are displayed under the **Topics** folder as well as for V7 queue managers.

Note: For details about SupportPac MS0Q refer to the IBM web page at:

`http://www.ibm.com/support/docview.wss?rs=171&uid=swg24013508&loc=en_US&cs=utf-8&lang=en`

8.2 Working with new properties and parameters

This section describes how to work with the new MQ object properties and parameters. The WebSphere MQ Explorer and MQSC command are covered.

Important: The only short explanation and parameter names for MQ Explorer and MQSC commands are written. For details about parameters behavior and setting refer to the MQ Explorer context help or to the topic *The MQSC commands*, in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.flat.doc/sc10340_.htm`

To use PCF command, refer to the topic Using Programmable Command Formats in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.flat.doc/pc10360_.htm`

Wildcards consideration:

On platforms other than z/OS, a string containing no characters (that is, two single quotation marks with no space in between) is interpreted as a quoted blank space, in other words, ('') is interpreted in the same way as (' '). The exception to this is if the attribute being used is one of the following:

- TOPICSTR
- SUB
- USERDATA
- SELECTOR

For the four objects listed above are two single quotation marks with no space interpreted as a zero-length string.

On z/OS the regular quoted blank space (' ') is needed. A string containing no characters ('') is the same as entering (), which is not valid.

8.2.1 Queue manager parameters

The following queue manager parameters have been added:

Time interval for browsing message

The queue manager might automatically unmark messages, that have been marked as browsed for the cooperating set of handles. If message is not processed within defined time interval, it is marked as not browsed.

To set up this parameter perform:

- In MQ Explorer: use parameter called

Author Comment: This parameter was not integrated in MQ Explorer at the time of writing.

  when change or display queue manager properties.
- With MQSC command: issue commands ALTER QMGR or DISPLAY QMGR with parameter MARKINT.

The parameter value is integer in milliseconds, in the range 0 through 999 999 999, default is 5000. There is a special value NOLIMIT which means that messages are not automatically unmarked.

Maximum size for message properties

The maximum length of properties data in bytes that can be associated with a message.

To set up this parameter perform:

- In MQ Explorer: use parameter called **Max properties length** in the **Extended** page when change or display queue manager properties.
- With MQSC command: issue commands ALTER QMGR or DISPLAY QMGR with parameter MAXPROPL.

The parameter value is integer in bytes, in the range 0 through 104 857 600 bytes (100 MB), default is NOLIMIT. Special value NOLIMIT means that there is no restriction for message properties.

Log defaults changes

Two changes have been made at logging parameters which can be set up for queue managers (by control command or in qm.ini and mqs.ini files):

- LogFilesPages default is now 4096. It was 1024 in the previous version of WebSphere MQ.
- LogBufferPages default is now 512. It was 128 in the previous version of WebSphere MQ.

8.2.2 Queue properties

The following queue parameters have been added:

Queue Alias

WebSphere MQ V7.0 introduces an extension to the ALIAS QUEUE object that allows an alias queue to be mapped to a topic object. The existing parameter, previously used as the pointer to the destination queue, has been renamed. The new parameter has been add to set up type of alias queue (queue or topic).

The details about using queue alias for topics are covered in the section 4.2.3, “Topic Alias” on page 49.

To set up these parameters perform:

- In MQ Explorer: new parameter is called **Base type** and the parameter value can be **Queue** or **Topic**. The renamed parameter is called **Base object** (instead of **Base Queue** in the previous version of WebSphere MQ) and the parameter value is the name of destination object (queue or topic).

  Both parameters are in the **General** page when define, change or display queue alias properties.
- With MQSC command: new parameter is called TARGTYPE and the parameter value can be QUEUE or TOPIC. The renamed parameter is called TARGET (instead of TARGQ in the previous version of WebSphere MQ) and the parameter value is the name of destination object (queue or topic).

  Issue commands DEFINE QALIAS, ALTER QALIAS or DISPLAY QALIAS with parameters TARGTYPE and TARGET.

  Note: The name TARGQ is retained for compatibility with your existing programs as parameter alias for TARGET for all MQSC QALIAS commands.

Message Properties

The message properties control attribute has been add. This parameter determines how message properties are delivered to getting applications and is applicable to local, alias and model queues.

The details about message properties are covered in the section 6.2, “Message properties” on page 91.

To set up this parameter perform:

- MQ Explorer: use parameter called **Property control** in the **Extended** page when define, change or display queue properties.
- MQSC command: issue commands DEFINE queues, ALTER queues or DISPLAY queues with parameter PROPCTL.

The parameter value can be:

- COMPAT (**Compatibility** in MQ Explorer)

  All message properties prefixed by mcd., jms., usr., or mqext. is delivered in MQRFH2. All other message properties are discarded. This is the default.
- ALL (**All** in MQ Explorer)

  All message properties included in one or more MQRFH2 headers.
- FORCE (**Force MQRFH2** in MQ Explorer)

  Message properties are always returned in MQRFH2 header, regardless of whether the application specifies a message handle. Properties is not accessible by message handle.
- NONE (**None** in MQ Explorer)

  All message properties are removed from the message.

Read ahead

Specifies the default read ahead behavior for non-persistent messages delivered to the client. Enabling read ahead can improve the performance of client applications consuming non-persistent messages. This parameter is applicable to local, alias and model queues.

MQ Explorer V7.0 can take advantage of this for remote queue managers administration (read ahead is set for SYSTEM.MQEXPLORER.REPLY.MODEL queue).

The details about read ahead feature are covered in the section 5.4, “Read ahead” on page 70.

To set up this parameter perform:

- MQ Explorer: use parameter called **Default read ahead** in the **Extended** page when define, change or display queue properties.
- MQSC command: issue commands DEFINE queues, ALTER queues or DISPLAY queues with parameter DEFREADA.

The parameter value can be:

- NO (**No** in MQ Explorer)

  Non-persistent messages are not read ahead unless the client defines it in the MQOPEN call. This is the default.
- YES (**Yes** in MQ Explorer)

  Non-persistent messages are sent to the client before the application requests them.
- DISABLED (**Disabled** in MQ Explorer)

  Messages are not sent ahead, regardless of the client application's request.

Asynchronous put

Specifies the behavior to be used by applications when the put response type, within the MQPMO options, is set to MQPMO_RESPONSE_AS_Q_DEF. This parameter is applicable to local, alias, model and remote queues.

The details about asynchronous put are covered in the section 5.5, "Asynchronous put" on page 73.

To set up this parameter perform:

- MQ Explorer: use parameter called **Default put response type** in the **Extended** page when define, change or display queue properties.
- MQSC command: issue commands DEFINE queues, ALTER queues or DISPLAY queues with parameter DEFPRESP.

The parameter value can be:

- SYNC (**Sync** in MQ Explorer)

  Ensures that the put operations to the queue are issued in synchronous mode (as if MQPMO_SYNC_RESPONSE option had been specified by application instead). This is the default.
- ASYNC (**Async** in MQ Explorer)

  Ensures that the put operations to the queue are issued in synchronous mode (as if MQPMO_ASYNC_RESPONSE option had been specified by application instead).

8.2.3 New channel and client connection properties

The following channel and client connection parameters have been add:

Message properties

The message properties control attribute has been add. This parameter determines how the messages are sent to pre-V7 queue managers. This parameter is applicable to sender, server, cluster sender and cluster receiver channels.

The details about message properties are covered in the section 6.2, “Message properties” on page 91.

To set up this parameter perform:

- MQ Explorer: use parameter called **Property control** in the **Extended** page when define, change or display channel properties.
- MQSC command: issue commands DEFINE CHANNEL, ALTER CHANNEL or DISPLAY CHANNEL with parameter PROPCTL.

The parameter value can be:

- COMPAT (**Compatibility** in MQ Explorer)

  If the message contains a property with a prefix of mcd., jms., usr. or mqext., all message properties are delivered to the application in an MQRFH2 header. Otherwise all properties of the message, except those contained in the message descriptor (or extension), are discarded and are no longer accessible to the application. This is the default value.
- ALL (**All** in MQ Explorer)

  All properties of the message are included with the message when it is sent to the remote queue manager.
- NONE (**None** in MQ Explorer)

  All properties of the message, except those in the message descriptor (or extension), are removed from the message before the message is sent to the remote queue manager.

Server-connections resources control

It is now possible to control server-connection resources to limit the number of connected clients or prevent a single workstation to monopolize our systems, because:

- This could be caused by a program bug
- Or by a Denial of Service (DoS) program
- Or by a unplanned roll-out of MQ Client application

The details about server-connection resources control are covered in the section 5.6, “Instance limits on SVRCONN channels” on page 77.

Maximum number of instances of server-connection channel

The maximum number of simultaneous instances of an individual server-connection channel that can be started.

To set up this parameter perform:

- In MQ Explorer: use parameter called **Max instances** in the **Extended** page when define, change or display server-connection channel properties.
- With MQSC command: issue commands DEFINE CHANNEL, ALTER CHANNEL or DISPLAY CHANNEL with parameters CHLTYPE(SVRCONN) and MAXINST.

The parameter value is integer, in the range 0 through 999 999 999, default is 999 999 999. A value 0 prevents all client access on this channel.

Maximum number of instances from a single client

The maximum number of simultaneous individual server-connection channels that can be started from a single client (detected by IP address).

To set up this parameter perform:

- In MQ Explorer: use parameter called **Max instance per client** in the **Extended** page when define, change or display server-connection channel properties.
- With MQSC command: issue commands DEFINE CHANNEL, ALTER CHANNEL or DISPLAY CHANNEL with parameters CHLTYPE(SVRCONN) and MAXINSTC.

The parameter value is integer, in the range 0 through 999 999 999, default is 999 999 999. A value 0 prevents all client access on this channel.

New error messages

When your client connections reach the limits the following error messages are written to the MQ error log and on Windows operating system also to Windows Event log:

- AMQ9489: The maximum number of instances was reached.

  The example of this error message is illustrated below:

Example 8-1 Example of AMQ9489 error message

```
12/5/2007 15:09:06 - Process(128.54) User(stora) Program(amqrmppa.exe)
AMQ9489: The maximum number of instances, 3, of channel 'QMHQ_STORES'
was reached.

EXPLANATION:
```

```
The server-connection channel 'QMHQ_STORES' is configured so that the
maximum number of instances that can run at the same time is 3. This
limit was reached.
ACTION:
Try the operation again when a new instance can be started.

If the limit has been reached because there are too many connections
from one or more of your client applications, consider changing the
applications to make fewer connections.

If you are not making use of sharing conversations, consider switching
to this mode of operation because several client connections can then
share one channel instance.
```

- AMQ9490: The maximum number of instances was reached for an individual client.

  The example of this error message is illustrated below:

Example 8-2 Example of AMQ9490 error message

```
12/5/2007 15:10:53 - Process(128.56) User(stora) Program(amqrmppa.exe)
AMQ9490: The maximum number of instances, 3, of channel 'QMHQ_STORES'
was reached for an individual client.

EXPLANATION:
The server-connection channel 'QMHQ_STORES' is configured so that the
maximum number of instances that can run at the same time for any
individual client is 3. This limit was reached for the client with
remote network address '192.168.16.76'.
ACTION:
Try the operation again when a new instance can be started for this
client.

If the limit has been reached because there are too many connections
from the relevant client application, consider changing the application
to make fewer connections.

If you are not making use of sharing conversations, consider switching
to this mode of operation because several client connections can then
share one channel instance.
```

Note: The error message AMQ9492 is also written to notice that there is a TCP/IP responder program error. This message is only pointer to the messages AMQ9489 and AMQ9489 which describe real reason of error.

Conversations sharing

WebSphere MQ V7.0 now supports conversations sharing for client connections. The separate TCP/IP socket had to be used for each connection in the previous version of WebSphere MQ. Now it is possible to set up number of connection that can be shared on each TCP/IP channel instance (on one TCP/IP socket). This parameter is applicable to server-connection channel and client connections definitions.

The details about conversations sharing are covered in the section 5.3, “Conversation sharing” on page 66.

Setting the conversations sharing limit

To set up this parameter perform:

- In MQ Explorer: use parameter called **Sharing Conversations** in the **Extended** page when define, change or display server-connection channel or client connections properties.
- With MQSC command: issue commands DEFINE CHANNEL, ALTER CHANNEL or DISPLAY CHANNEL with parameters CHLTYPE(SVRCONN) or (CLNTCONN) and SHARECNV.

The parameter value is integer, in the range 0 through 999 999 999, default is 10. And there are two special values:

- 1

  Sharing of conversations is disabled. Channel runs in V7 mode and all new connection features are still available.
- 0

  Sharing of conversations is disabled. Channel runs in pre-V7 mode and the read ahead and client asynchronous consume features are not available.

Displaying the conversations sharing parameters

To display the conversations sharing limit and the current number of conversations currently running over that channel instance.

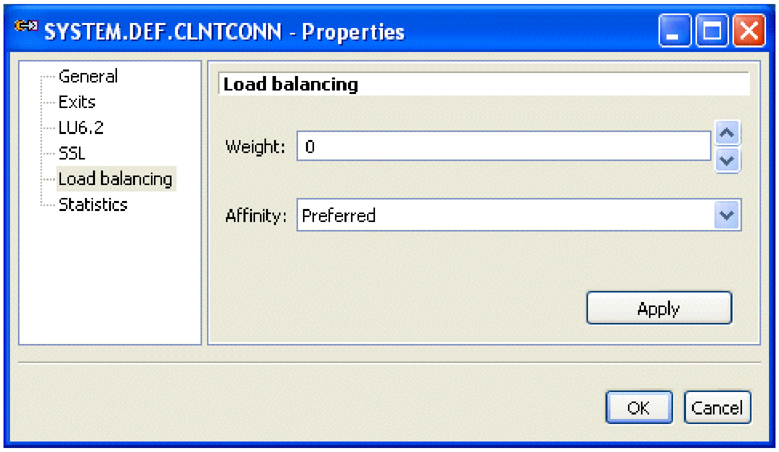
To display these parameters perform:

- In MQ Explorer: right-click to server-connection channel and select **Status** → **Channel Status...** then refer to the values **Max Conversations** and **Current Conversations** (the last two columns in the list).
- With MQSC command: issue commands DISPLAY CHSTATUS with parameters MAXSHCNV and CURSHCNV.

The 0 (zero) values for these parameters means that channel is running in pre-V7 mode.

Client connection load balancing

WebSphere MQ V7.0 now supports client connection load balancing. There are two new parameters to set up load balancing feature. In case of MQ Explorer both parameters can be set up from the new **Load balancing** page when working with client connections wizard, as illustrated at the figure below:



Weight: 0

Affinity: Preferred

Figure 8-18 Client connection load balancing

The details about client connection load balancing are covered in the section 5.7, “Weighted selection on CLNTCONN channels” on page 80.

Client connection affinity

The channel affinity attribute is used so client applications that connect multiple times using the same queue manager name can choose whether to use the same client channel definition for each connection. This parameter is only applicable when multiple channel definitions are available.

To set up this parameter perform:

- MQ Explorer: use parameter called **Affinity** in the **Load balancing** page when define, change or display client connection properties.
- MQSC command: issue commands DEFINE CHANNEL, ALTER CHANNEL or DISPLAY CHANNEL with parameters CHLTYPE(CLNTCONN) and AFFINITY.

The parameter value can be:

- PREFERRED (**Preferred** in MQ Explorer)

  The first connection in a process reading a CCDT creates a list of applicable definitions based on the weighting with any applicable CLNTWGHT(0) definitions first and in alphabetical order. Each connection in the process attempts to connect using the first definition in the list. If a connection is unsuccessful the next definition is used. Unsuccessful non CLNTWGHT(0) definitions are moved to the end of the list. CLNTWGHT(0) definitions remain at the start of the list and are selected first for each connection. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created. Each client process with the same hostname creates the same list. This is the default option.
- NONE (**None** in MQ Explorer)

  The first connection in a process reading a CCDT creates a list of applicable definitions. All connections in a process select an applicable definition based on the weighting with any applicable CLNTWGHT(0) definitions selected first in alphabetical order. For C, C++ and .NET (including fully managed .NET) clients the list is updated if the CCDT has been modified since the list was created.

Client connection weight

The client channel weighting attribute is used so client channel definitions can be selected at random based on their weighting when more than one suitable definition is available.

To set up this parameter perform:

- MQ Explorer: use parameter called **Weight** in the **Load balancing** page when define, change or display client connection properties.
- MQSC command: issue commands DEFINE CHANNEL, ALTER CHANNEL or DISPLAY CHANNEL with parameters CHLTYPE(CLNTCONN) and CLNTWGHT.

The parameter value is integer, in the range 0 through 99, default is 0. The special value 0 indicates that no random load balancing is performed and applicable definitions are selected in alphabetical order.

8.2.4 The connection status enhancements

It is now possible to display connection information about the applications connected to the queue manager or queue. This is a useful because it enables you to identify which applications are connected and can be use, for example, to detect long-running units of work.

The following connection status parameters have been add:

Display connection status for read ahead

This command shows the state of read ahead capability on a connection handle.

The details about asynchronous consumption (call back) are covered in the section 5.4, “Read ahead” on page 70.

To display connection parameters perform:

- In MQ Explorer: right-click to queue manager and select **Application Connections...** then in the status window refer to the value **Read ahead** at the lower section of this window.
- With MQSC command: issue command DISPLAY CONN with parameter READA.

The displayed parameter values can be:

- NO (**No** in MQ Explorer)

  Read ahead of non-persistent messages is not enabled for this object.
- YES (**Yes** in MQ Explorer)

  Read ahead of non-persistent message is enabled for this object and is being used efficiently.
- INHIBITED (**Inhibited** in MQ Explorer)

  Read ahead was requested by the application but has been inhibited because of incompatible options specified on the first MQGET call.
- BACKLOG (**Backlog** in MQ Explorer)

  Read ahead of non-persistent messages is enabled for this object. Read ahead is not being used efficiently because the client has been sent a large number of messages which are not being consumed.

Display connection status for asynchronous consume

These command shows the state of asynchronous consumption on a connection handle.

The details about asynchronous consumption (call back) are covered in the section 6.4, “Call Back for asynchronous consumers” on page 109.

At a queue manager

To display connection parameters perform:

- In MQ Explorer: right-click to queue manager and select **Application Connections...** then in the status window refer to the value **Asynchronous state** at the lower section of this window.
- With MQSC command: issue command DISPLAY CONN with parameter ASTATE.

The displayed parameter values can be:

- SUSPENDED (**Suspended** in MQ Explorer)

  The asynchronous message consumption is temporarily suspended on this connection - an MQCTL call with MQOP_SUSPEND has been issued against the connection handle.
- STARTED (**Started** in MQ Explorer)

  The asynchronous message consumption can proceed on this connection - an MQCTL call with MQOP_START has been issued against the connection handle.
- STARTWAIT (**Startwait** in MQ Explorer)

  The asynchronous message consumption can proceed on this connection - an MQCTL call with MQOP_START_WAIT has been issued against the connection handle.
- STOPPED (**Stopped** in MQ Explorer)

  The asynchronous message consumption cannot currently proceed on this connection - an MQCTL call with MQOP_STOP has been issued against the connection handle.
- NONE (**None** in MQ Explorer)

  The asynchronous message consumption cannot currently proceed on this connection - no MQCTL call has been issued against the connection handle.

At a queue

To display connection parameters perform:

- In MQ Explorer: right-click to queue and select **Status...** then in the status window refer to the value **Asynchronous state** at the lower section of this window.
- With MQSC command: issue command DISPLAY QSTATUS with parameter ASTATE.

The displayed parameter values can be:

- ACTIVE (**Active** in MQ Explorer)

The asynchronous message consumption is running on this object handle - an MQCB call has set up a function to call back to process messages asynchronously.

- INACTIVE (**Inactive** in MQ Explorer)

  The connection handle has not yet been started, or has been stopped or suspended - an MQCB call has set up a function to call back to process messages asynchronously.

- SUSPENDED (**Suspended** in MQ Explorer)

  The asynchronous message consumption is suspended on this object handle - this can be either because an MQCB call with MQOP_SUSPEND has been issued by the application, or because it has been suspended by the system.

- SUSPTEMP (**Susptemp** in MQ Explorer)

  The asynchronous message consumption is temporarily suspended by the system.

- NONE (**None** in MQ Explorer)

  The asynchronous message consumption cannot currently proceed on this connection - no MQCTL call has been issued against the connection handle.

8.3 Java and JMS related administration enhancements

WebSphere MQ V7.0 offers these new administration capabilities for application developers:

- Embedded PCF support for Java
- WebSphere MQ classes for JMS has been enhanced to provide higher level of serviceability.

8.3.1 Embedded PCF support for Java

The support for creating WebSphere MQ message headers and messages containing PCF commands is now contained in the file com.ibm.mq.headers.jar which is part of standard WebSphere MQ V7.0 product installation.

This capability is available for previous versions of WebSphere MQ with SupportPac *MS0B: WebSphere MQ Java classes for PCF*.

Note: The Programmable Command Format (PCF) commands allow to write administration tasks into the a program (either administration or application program). So you can create queues and process definitions, and change queue managers, from this program.

PCF commands cover the same range of functions provided by MQSC commands.

For detailed information, refer to the topic *Administration using PCF commands* in the WebSphere MQ V7.0 Information Center at:

https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.flat.doc/ia11070_.htm

8.3.2 WebSphere MQ classes for JMS

WebSphere MQ classes for JMS has been enhanced to provide higher level of serviceability:

- Tracing: an application can start and stop tracing, specify the required level of detail in a trace and customize trace output in various ways.
- Logging: WebSphere MQ classes for JMS maintains a log file, which contains messages about errors that need to be correct (the messages are written in plain text). A JMS application can specify the location of the log file and its maximum size.
- First Failure Support Technology™ (FFST™): support of the FFST technique is now available for JMS applications. The FFST report contains information that IBM Service can use to diagnose the problem more quickly.
- Version information: an application can query the version of WebSphere MQ classes for JMS.
- Exception messages: exception messages have been enhanced to provide more information about the causes of errors and the actions required to correct errors.
- Application servers: the integration of the serviceability features of WebSphere MQ classes for JMS with WebSphere Application Server has been improved.

8.4 The control commands

This section contains the new control commands or commands with new parameters.

Important: The only short explanation and parameter names for control commands are written. For details about parameters behavior and setting refer to the topic *The control commands*, in the WebSphere MQ V7.0 Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp?topic=/com.ibm.mq.flat.doc/fa15550_.htm`

8.4.1 Create queue manager (crtmqm)

The **`crtmqm`** command has three new options, available on WebSphere MQ for Windows only:

- **`-sa`**: automatic queue manager startup.

  The queue manager is configured to start automatically when the machine starts up and runs as a service. This is the default option when create queue manager from MQ Explorer.

- **`-si`**: interactive (manual) queue manager startup.

  The queue manager is configured to start only when manually issue **`strmqm`** command, then runs under the logged on (interactive) user and ends when the user who started them logs off.

- **`-ss`**: service (manual) queue manager startup.

  The queue manager is configured to start only when manually issue **`strmqm`** command, then runs as a service and continue to run even after the interactive user has logged off. This is the default option when create queue manager from command line.

The command syntax for the new parameters can be:

- **`crtmqm -sa`** ***`Queue manager_name`***
- **`crtmqm -si`** ***`Queue manager_name`***

8.4.2 Start queue manager (strmqm)

The **`strmqm`** command has two new options, available on WebSphere MQ for Windows only:

- `-si`: interactive (manual) queue manager startup.

  The queue manager runs under the logged on (interactive) user and ends when the user who started them logs off.
- `-ss`: service (manual) queue manager startup.

  The queue manager runs as a service and continue to run even after the interactive user has logged off.

These parameters override any startup type set previously by the **`crtmqm`** command, the **amqmdain** command, or the MQ Explorer. If not specify any parameter, the startup type set up when create queue manager is used.

The command syntax for the new parameters can be:

- `strmqm` ***`Queue manager_name`*** `-si`
- `strmqm` ***`Queue manager_name`*** `-ss`

8.4.3 Start HTTP listener (?command?)

Author Comment: At the time of writing the first version of the book there was no information about integrated HTTP listener

8.5 Journal time stamps on i5/OS

WebSphere MQ V7.0 for i5/OS uses a different time stamping technique from earlier versions of WebSphere MQ for iSeries. This avoids the problem of duplicate time stamps for journals at the end of daylight saving time. You no longer need to take any special action after setting the clock back.

9

Publish/Subscribe management

The WebSphere MQ v7.0 Publish/Subscribe environment can be managed using the MQ Explorer or MQSC. This chapter shows how to use these utilities provided in WebSphere MQ v7.0 to setup your Publish/Subscribe Environment. It covers the following topics:

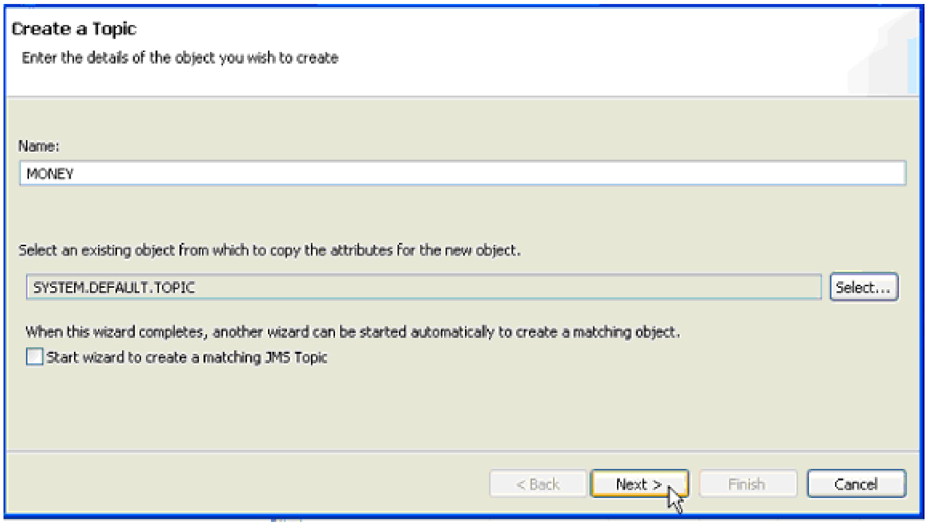
- “Creating Topics using MQ Explorer” and “Creating Topics using MQSC”
- “Altering Topics using MQ Explorer” and “Altering Topics using MQSC”
- “Displaying Topic Status using MQ Explorer” and “Displaying Topic Status using MQSC”
- “Creating JMS Topics using MQ Explorer”
- “Setting up Topic Security using MQ Explorer”and “Setting up Topic Security using setmqaut”
- “Mapping Queue Aliases to a Topic Object using MQSC”
- “Managing Subscriptions”

9.1 Managing Topics

This section illustrates the usage of MQ Explorer and MQSC commands for managing topics in your Publish/Subscribe Environment

9.1.1 Creating Topics using MQ Explorer

- Start the MQ Explorer either using the Desktop Icon or using the Start Menu.
- Once the explorer is started, navigate to the "Topics" folder in the WMQ7 Queue Manager
- From the Navigation Menu select "Topics" under queue manager WMQ7, Right Mouse-click and select "New" - "Topic" from the context menu


Create a Topic

Enter the details of the object you wish to create

Name:
MONEY

Select an existing object from which to copy the attributes for the new object.

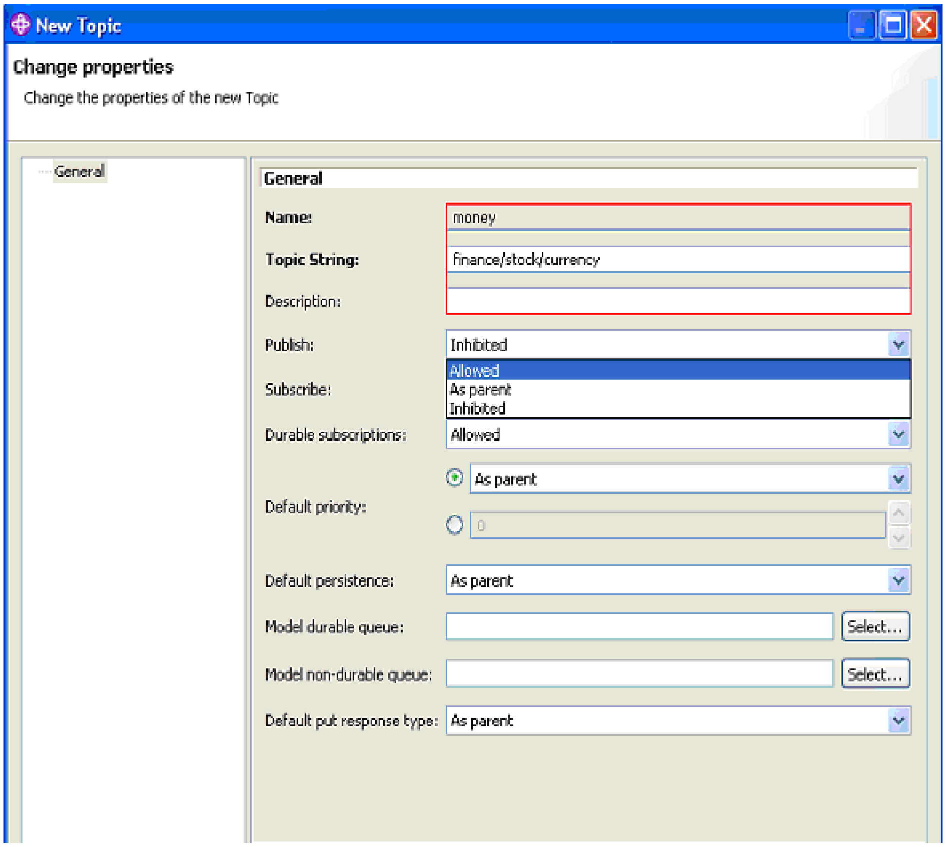
SYSTEM.DEFAULT.TOPIC

When this wizard completes, another wizard can be started automatically to create a matching object.

☐ Start wizard to create a matching JMS Topic

Figure 9-1 Create a Topic

- The Create a Topic dialog box appears
- A name for the topic object that needs to be created can be entered in the Name field and then clicking on Next leads to the Change Properties dialog



Name: money
Topic String: finance/stock/currency
Description:
Publish: Inhibited
Allowed
As parent
Inhibited
Subscribe:
Durable subscriptions: Allowed
Default priority: [✓ As parent] 0
Default persistence: As parent
Model durable queue:
Model non-durable queue:
Default put response type: As parent

Figure 9-2 Topic Creation - Change Properties

- A Change Properties Dialog appears. A topic string needs to be entered in the Topic String text field. This Topic String depicts the Topic Hierarchy. It is optional to enter a description for the Topic.

  The Publish property can take on three values: Allowed, As Parent and Inhibited

  Allowed: Suitably authorized applications can publish to this topic.

  As Parent: Whether messages can be published to the topic is based on the setting of the closest parent administrative topic object in the topic tree. This is the default supplied with WebSphere MQ, however this can be changed administratively.

  Inhibited: Messages cannot be published to this topic. The default is As Parent.

The Subscribe property can also take on three values: Allowed, As Parent and Inhibited. These values mean the same thing as they do in the Publish property; except for it is in the context of subscription. The default is As Parent.

The Durable Subscriptions Property can take on three values: Allowed, As Parent and Inhibited, where Allowed is the default.

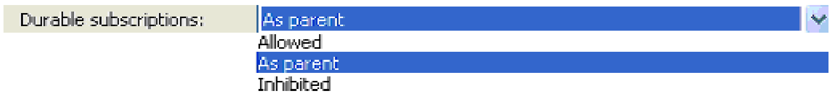

Durable subscriptions: As parent
Allowed
As parent
Inhibited

Figure 9-3 Configuring Durable Subscriptions

As Parent: Whether durable subscriptions can be made on this topic is based on the setting of the closest parent administrative topic object in the topic tree. This is the default supplied with WebSphere MQ, however this can be changed administratively.

Allowed: Durable subscriptions can be made on this topic.

Inhibited: Durable subscriptions cannot be made on this topic.

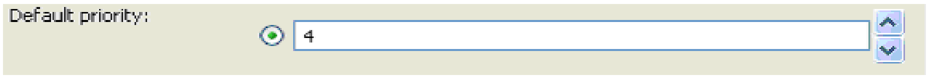

Default priority: 4

Figure 9-4 Configuring Default Priority

The Default Priority value can be set to assign a default priority on messages published to that topic. The value must be in the range zero (the lowest priority), through to the MAXPRTY queue manager parameter (MAXPRTY is 9). If Default Priority is set to As Parent, then the default priority is based on the setting of the closest parent administrative topic object in the topic tree. This is the default supplied with WebSphere MQ, but can be changed administratively.

The Default Persistence field specifies the message persistence that applications require.

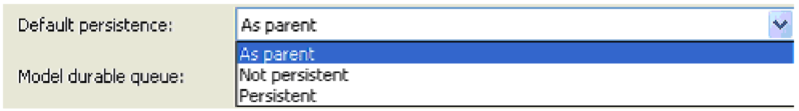

Default persistence: As parent
Model durable queue:
As parent
Not persistent
Persistent

Figure 9-5 Configuring Default Persistence

As Parent means that the default persistence is based on the setting of the closest parent administrative topic object in the topic tree. This is the default supplied with WebSphere MQ, however this can be changed administratively.

Not Persistent means that messages are lost during a restart of the queue manager. Persistent means that messages survive a queue manager restart

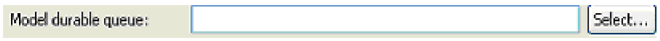


Figure 9-6 Configuring the Model Durable Queue

The Model Durable Queue property can be used for durable subscriptions that request that the queue manager manages the destination of its publications. The maximum length for this property is 48 characters. If this property is blank, it operates in the same way as As Parent values on other properties. The name of the model queue to be used is based on the closest parent administrative topic object in the topic tree with a value set for this property.

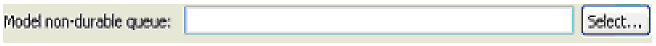


Figure 9-7 Configuring the Model non-durable Queue

The Model Non-Durable Queue property can be used for non-durable subscriptions that request that the queue manager manages the destination of its publications. The maximum length for this property is 48 characters. If this property is blank, it operates in the same way as As Parent values on other properties. The name of the model queue to be used is based on the closest parent administrative topic object in the topic tree with a value set for this property.

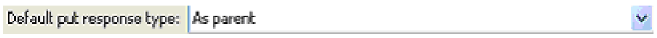

Default put response type: As parent

Figure 9-8 Configuring Default-Put Response Type

The Default Put Response Type property is a way of administratively configuring the Topic Object to support Synchronous/Asynchronous mechanisms for responding to Message Puts (MQPUT) from WebSphere MQ Clients. WebSphere MQ v6.0 supports the synchronous put response type. This means that every MQPUT call is synchronously responded with a completion code. WebSphere MQ v7.0 introduces the Asynchronous put response type (selecting Async). Further detail on concepts involved and using this feature in WebSphere MQ v7.0 are discussed in Chapter 6, “Message Queue Interface extensions” on page 87. The As Parent attribute is the default value for this property and has the same meaning as it has on all other properties.

Having configured all these properties out of which only the Topic Name and the Topic String are required and all other are optional you can then click on Finish. Dismiss the confirmation box by clicking on `OK`.

9.1.2 Creating Topics using MQSC

- Open a command window using the command window icon on the desktop and type.Type RUNMQSC
- Type the command **`DEFINE TOPIC(DELI) TOPICSTR('store')`** . This command creates a topic by the name DELI

  Try typing the command **`DEFINE TOPIC (DELI.FRESH) TOPICSTR ('deli/fresh')`**. This creates a topic object DELI.FRESH which is a child of the topic object DELI created previously and thus, inherits all attributes from the same.

  Try typing the command **`DEFINE TOPIC (STORE) TOPICSTR ('store')`**. This results in an error saying A WebSphere MQ Topic using the supplied topic string already exists. This is because this topic definition conflicts with the definition of the topic object DELI. This also means that two administrative topics cannot be created at the same node level in the topic hierarchy. .

  The new parameter options available for the **`DEFINE TOPIC`** command in MQSC are the same as those discussed on the Change Properties Dialog while creating topics on WebSphere MQ v7.0 Explorer. However, they have different names in the MQSC syntax. A mapping between the MQSC attributes and the corresponding MQ Explorer Properties is illustrated in the table below:

Table 9-1 Mapping of MQ Explorer Topic Creation Properties to MQSC DEFINE TOPIC Command Parameter Options

MQ Explorer Property	**MQSC Attribute**	
	Name	Possible Values
Publish	PUB	ASPARENT ENABLED DISABLED
Subscribe	SUB	ASPARENT ENABLED DISABLED
Durable Subscriptions	DURSUBS	ASPARENT YES NO
Default Priority	DEFPRTY	INTEGER FROM 0 TO 9
Default Persistence	DEFPSIST	ASPARENT YES NO

MQ Explorer Property	MQSC Attribute	
Model Durable Queue	MDURMDL	STRING
Model Non Durable Queue	MNDURMDL	STRING

- Besides the above-mentioned attributes, WebSphere MQ v7.0 introduces new attributes, which can be used with the **DEFINE TOPIC** command. Further detail on how to use these attributes can be found under the **DEFINE TOPIC** command listed under The MQSC Commands under Script MQSC Command Reference section of the WebSphere MQ v7.0 Infocenter.

9.1.3 Altering Topics using MQ Explorer

- On the MQ Explorer general view, click on **Topics** which displays the list of topics created in the system. Right click on a topic name and click on Properties.
- Clicking on **Properties** shows up the <Topic-name>-Properties Dialog box.

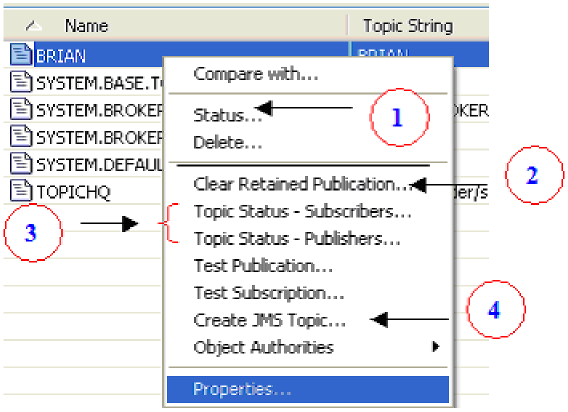
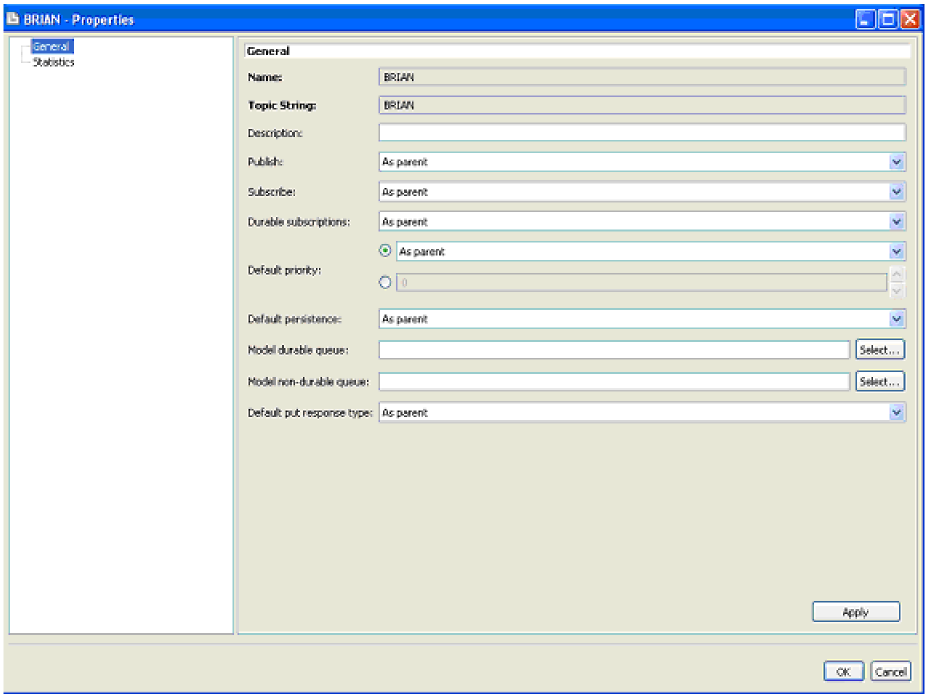


Figure 9-9 Topic Menu

- The properties listed here are the same as those listed on the Change Properties dialog during Topic Creation. If you choose to change these properties then click **Apply** and then **OK** for the changes to take effect.



General

Name: BRIAN

Topic String: BRIAN

Description:

Publish: As parent

Subscribe: As parent

Durable subscriptions: As parent

Default priority: [✓ As parent] 0

Default persistence: As parent

Model durable queue:

Model non-durable queue:

Default put response type: As parent

Figure 9-10 Changing Topic Properties

9.1.4 Altering Topics using MQSC

- WebSphere MQ v7.0 provides an **`ALTER TOPIC MQSC`** command that can be used in altering topics. The parameter options that can be used with this command are the same as those that can be used with the **`DEFINE TOPIC`** command.

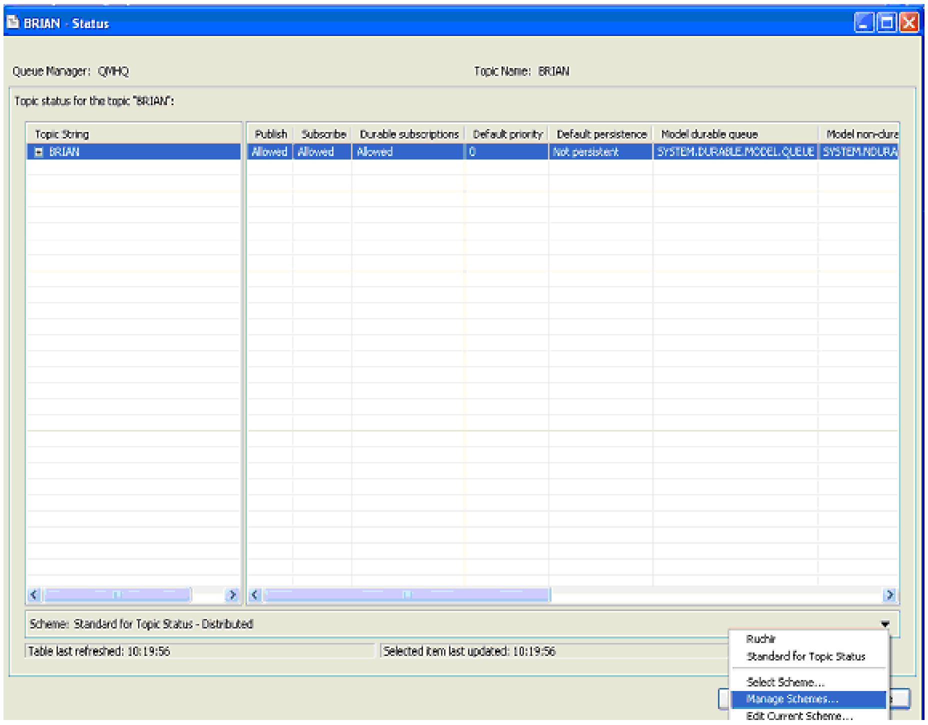
Example 9-1 Using ALTER TOPIC command

```
ALTER TOPIC (DELI) PUB (DISABLED) SUB (DISABLED)
```

9.1.5 Displaying Topic Status using MQ Explorer

- Right click on the topic name in MQ Explorer and then click on **Status** marked as 1 in Figure 9-9. This displays a window showing the current values for all the properties of the Topic. In case only a given subset of the properties needs to be displayed then WebSphere MQ v7.0 explorer provides an option of creating a scheme. Right click on the **Scheme Panel** and click on **Manage**

Schemes. You can tailor your own display scheme, which may only consist of the properties that you are interested in viewing.


BRIAN - Status

Queue Manager: QMHQ

Topic Name: BRIAN

Topic status for the topic "BRIAN":

Topic String	Publish	Subscribe	Durable subscriptions	Default priority	Default persistence	Model durable queue	Model non-dura
BRIAN	Allowed	Allowed	Allowed	0	Not persistent	SYSTEM.DURABLE.MODEL.QUEUE	SYSTEM.NDURA

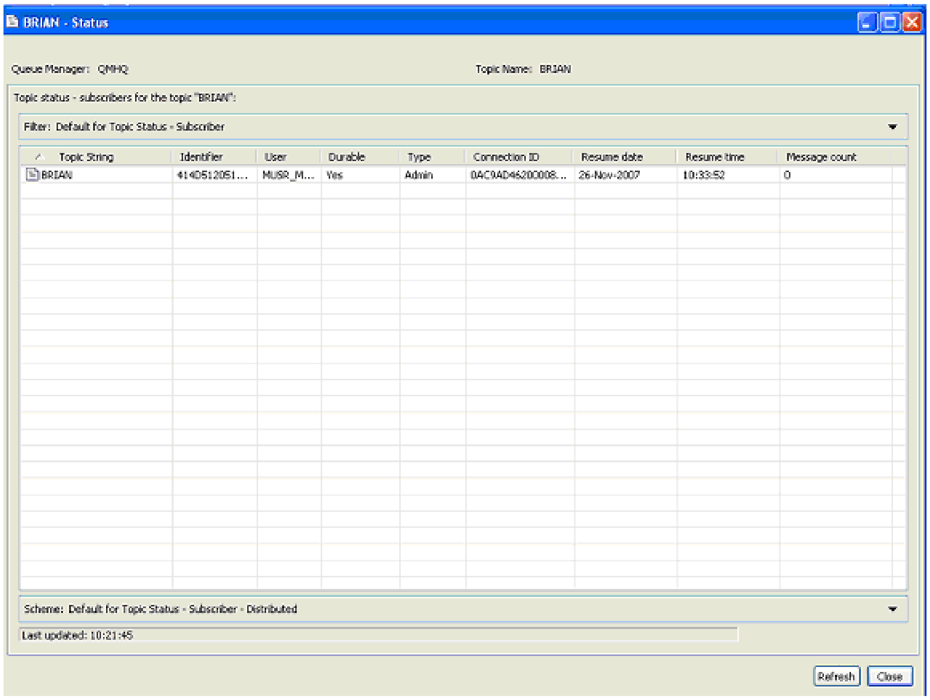
Scheme: Standard for Topic Status - Distributed

Table last refreshed: 10:19:56

Selected item last updated: 10:19:56

Figure 9-11 Displaying Topic Status

- Right click on the topic name in MQ Explorer and then click on **Topic Status - Publishers and Topic Status - Subscribers** marked as 3 in Figure 9-9. This opens a window displaying status of publications and subscriptions respectively. For example, clicking on Topic Status - Subscribers displays the window below:



Queue Manager: QMHQ Topic Name: BRIAN

Topic status - subscribers for the topic "BRIAN":

Filter: Default for Topic Status - Subscriber

Topic String	Identifier	User	Durable	Type	Connection ID	Resume date	Resume time	Message count
BRIAN	414D512051...	MUSR_M...	Yes	Admin	0AC9AD46200008...	26-Nov-2007	10:33:52	0

Scheme: Default for Topic Status - Subscriber - Distributed

Last updated: 10:21:45

Figure 9-12 Displaying Subscriber Status

9.1.6 Displaying Topic Status using MQSC

- The **`DISPLAY TPSTATUS`** command can be used to display the status of one or more topic nodes in a topic tree, as specified by the parameters supplied. This command can be specified as:

Example 9-2 Displaying Topic Status using TPSTATUS command

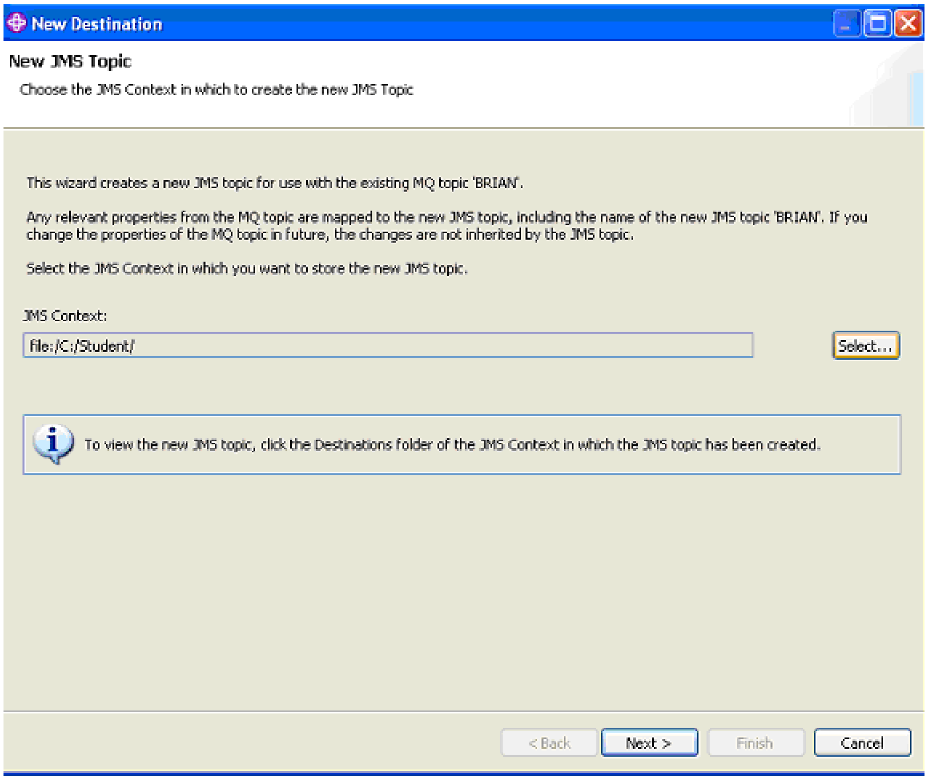
```
DISPLAY TPSTATUS TOPIC_STRING <WHERE> <FILTER_WORD> <OPERATOR>
<FILTER_VALUE> <ALL>
```

The DISPLAY TPSTATUS command requires a topic string value to determine which topic nodes the command returns. The WHERE keyword specifies a filter condition to display only those administrative topic definitions that satisfy the selection criterion of the filter condition. The filter condition is in three parts: filter-keyword, operator, and filter-value. The filter keyword can use any attribute except the CMDSCOPE attribute. List of all the other attributes that can be used here is found on the WebSphere MQ Infocenter under the DISPLAY TPSTATUS command described under The MQSC Commands under Script MQSC Command Reference section of the

WebSphere MQ v7.0 Infocenter. The parameter ALL can be used to display the values of all properties of a Topic. Please also note that displaying of Topic Status can be wild carded. For example, `DISPLAY TPSTATUS ('#')`. The hash (#) matches any topic string and therefore this command displays values of properties of topic objects corresponding to each topic string currently defined in the system.

9.1.7 Creating JMS Topics using MQ Explorer

Right click on the topic name in MQ Explorer and then click on **Create a JMS Topic** marked as 4 in Figure 9-9. This displays a dialog for creating a JMS Topic which is the JMS equivalent of the corresponding MQ Topic.


New Destination

New JMS Topic

Choose the JMS Context in which to create the new JMS Topic

This wizard creates a new JMS topic for use with the existing MQ topic 'BRIAN'.

Any relevant properties from the MQ topic are mapped to the new JMS topic, including the name of the new JMS topic 'BRIAN'. If you change the properties of the MQ topic in future, the changes are not inherited by the JMS topic.

Select the JMS Context in which you want to store the new JMS topic.

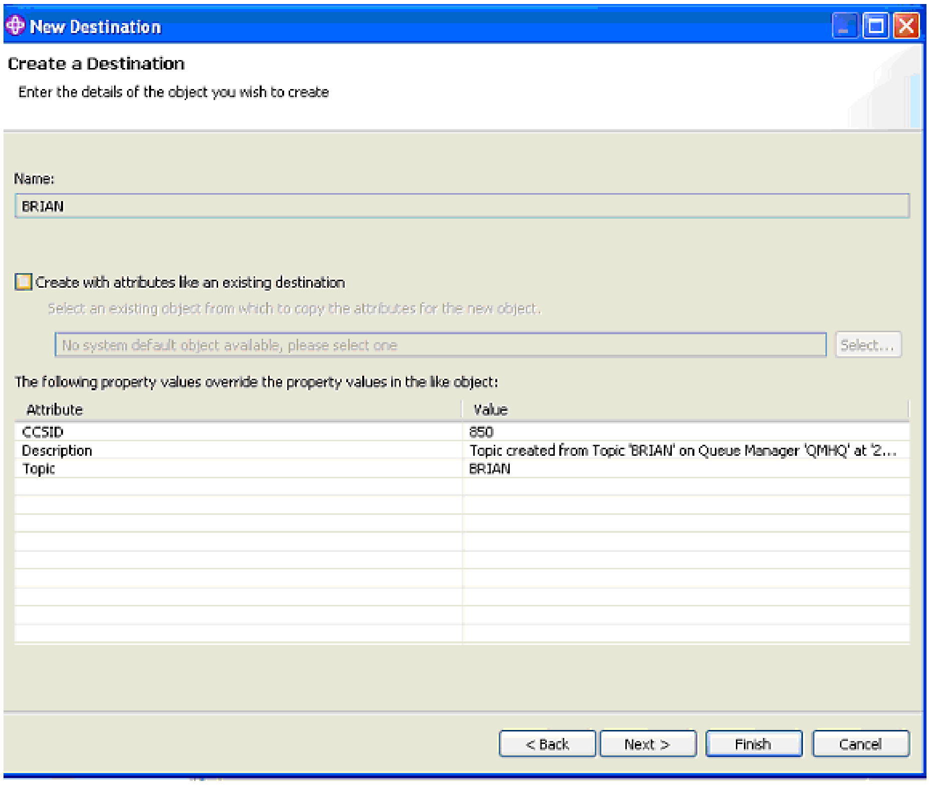
JMS Context: file:/C:/Student/

To view the new JMS topic, click the Destinations folder of the JMS Context in which the JMS topic has been created.

Figure 9-13 Creating a JMS Topic using MQ Explorer

An existing JMS context is a pre-requisite to creation of a JMS Topic. The JMS Context can be created by right clicking on the **JMS Administered Objects** in the right hand side tree view menu of MQ Explorer and selecting

Add Initial Context. The Initial Context thus created can be selected in the JMS Context field. Click Next.



Name: BRIAN

Create with attributes like an existing destination

Select an existing object from which to copy the attributes for the new object.

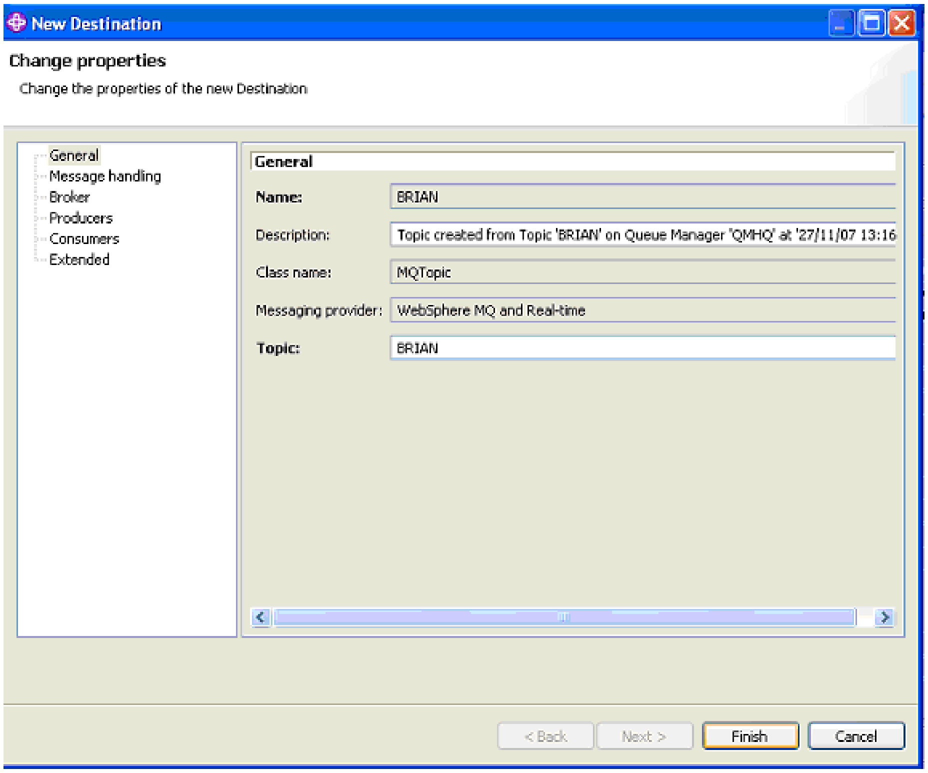
No system default object available, please select one

The following property values override the property values in the like object:

Attribute	Value
CCSID	850
Description	Topic created from Topic 'BRIAN' on Queue Manager 'QMHQ' at '2...
Topic	BRIAN

Figure 9-14 Creating a JMS Topic like a pre-existing JMS Topic

On this screen you can choose to select the Create with attributes like an existing definition in case you want that the attributes of the new JMS Topic that you are going to create get copied from a pre-existing default object definition. You can now either click Finish to complete the creation of the JMS Topic or else you can click Next to go to the Change Properties dialog box.



Name: BRIAN
Description: Topic created from Topic 'BRIAN' on Queue Manager 'QMHQ' at '27/11/07 13:16
Class name: MQTopic
Messaging provider: WebSphere MQ and Real-time
Topic: BRIAN

Figure 9-15 Change properties of the JMS Topic

On the right hand side tree view menu of the Change Properties dialog box, you see categories of properties that can be changed on the JMS Topic.

The Message Handling tab has properties such as Expiry, Persistence and Priority that has to do with how the way Messages are handled by MQ, as the name suggests. Expiry can have values Application which means it can be set inside the JMS Application, or Unlimited, which means messages have no expiry, or a user specified value.

Persistence can have values Application; meaning that it can be set inside the JMS Application, or High; meaning that irrespective of messages being persistent or non-persistent they survive a queue manager restart. If the NPMCLASS value for that particular queue is also set to High, Persistent and Non Persistent would mean the same as it does in WebSphere MQ v6 Queue Default; meaning that default persistence level is set for that particular queue which holds the messages. Priority means the same thing as does in WebSphere MQ v6.

The Broker tab has properties such as Broker Durable Subscription Queue, which points to the queue that hold messages for durable subscriptions for JMS based clients. The Broker CC Durable Subscription Manager Queue serves the same functionality as the Broker Durable Subscription Queue except for it caters to JMS 1.1 based clients. The Publication Stream refers to the Stream name. Streams provide a way of separating the flow of information for different topics. A stream is implemented as a set of queues, one at each broker that supports the stream. Each queue has the same name (the name of the stream). The default stream set up between all the brokers in a network is called SYSTEM.BROKER.DEFAULT.STREAM.

Note: The use of streams is deprecated in WebSphere MQ Version 7.0. In order to be backward compatible WebSphere MQ Version 7.0 produces topic objects and topic strings by combining WebSphere MQ Version 6.0 StreamName and Topic parameters. For example, if the WebSphere MQ Version 6.0 StreamName is MATT.RETAIL.CAT and the Topic String is Deli/Fresh/Fruit, the WebSphere MQ Version 7.0 publish/subscribe engine creates a topic called /MATT/RETAIL/CAT/Deli/Fresh/Fruit.

The Broker Publication Queue Manager maps to the Queue Manager on which the underlying MQ Topic is defined. Please note that there is a one to one mapping between JMS Topic properties and MQ Topic properties. However, the JMS Topic properties are not subject to change when the corresponding MQ Topic properties change.

The Producer tab has the property Asynchronous Puts that can have values As Destination, which means that the value of this property depends on the settings for Asynchronous Puts on the destination queue being used, or Enabled or Disabled which mean that Asynchronous Puts are administratively disabled or enabled.

The Consumer has the property Allow Read Ahead that can have values As Destination, Enabled or Disabled. These have the same meanings as that for Asynchronous Puts on the Producer tab.

9.1.8 Setting up Topic Security using MQ Explorer

Right click on the topic name in MQ Explorer and then click on **Object Authorities** and then click on **Manage Authority Records.**

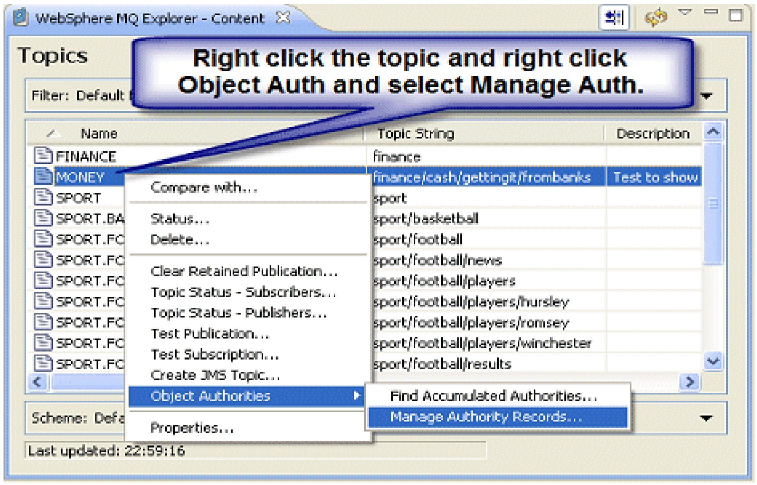
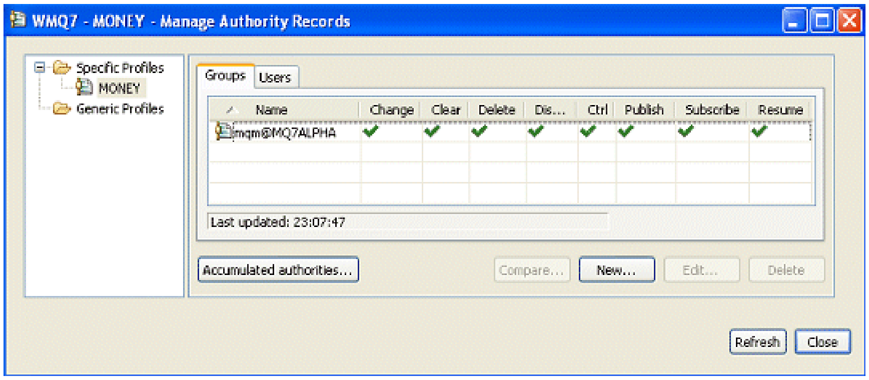

WebSphere MQ Explorer - Content
Topics
Right click the topic and right click Object Auth and select Manage Auth.
Filter: Default
Name
Topic String
Description
FINANCE
MONEY
SPORT
finance
finance/cash/gettingit/frombanks
Test to show
sport
sport/basketball
sport/football
sport/football/news
sport/football/players
sport/football/players/hursley
sport/football/players/romsey
sport/football/players/winchester
sport/football/results
Compare with...
Status...
Delete...
Clear Retained Publication...
Topic Status - Subscribers...
Topic Status - Publishers...
Test Publication...
Test Subscription...
Create JMS Topic...
Object Authorities
Properties...
Find Accumulated Authorities...
Manage Authority Records...
Scheme: Default
Last updated: 22:59:16

Figure 9-16 Open Object Authorities in MQ Explorer Navigator View

This opens the **Manage Authority Records** Dialog as shown below. An authority record is the set of authorities that have been granted to a particular user or group of users (entities) on a named object.



Name	Change	Clear	Delete	Dis...	Ctrl	Publish	Subscribe	Resume
mqm@MQ7ALPHA	✓	✓	✓	✓	✓	✓	✓	✓

Figure 9-17 Managing Specific Profiles

This dialog shows which groups have access to the topic.A specific profile applies only to the object of that name and type. To grant or revoke an authority on a single object, you select the relevant specific profile and create or edit the authority records for that profile. A generic profile matches one or more objects using wildcard characters.

Although this interface looks similar to that in WebSphere MQ v6.0, there are three new authorities that are now added here with respect to the

Publish/Subscribe Integration in WebSphere MQ v7.0. These three new authorities are:

Publish: Enabling this authority means that users of this group can publish to topic `MONEY` using the MQPUT call.

Subscribe: Enabling this authority means that users of this group can Create, alter or resume a subscription to a topic using the MQSUB call.

Resume: Enabling this authority means that users of this group can Resume a subscription using the MQSUB call.

You can even choose to create your own group authority or user authority as per your requirements as illustrated below.

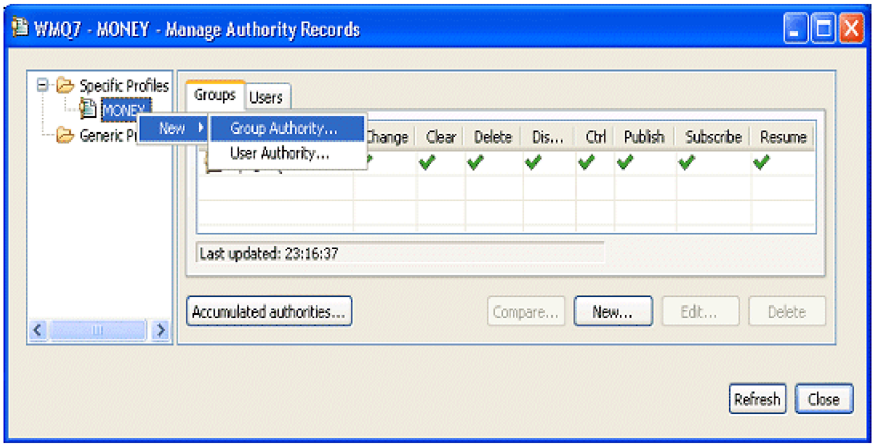
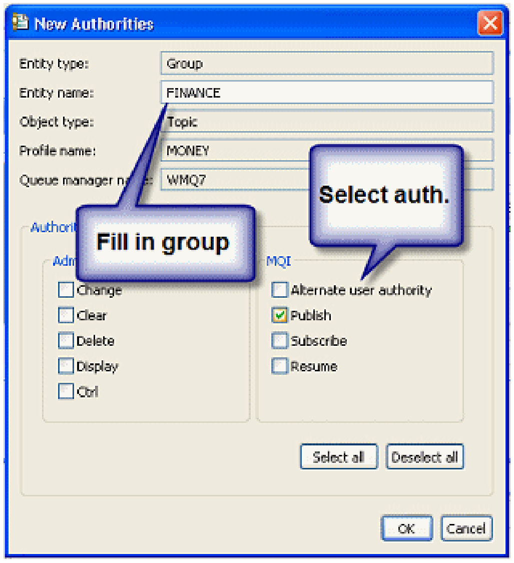


Figure 9-18 Creating GROUP Authority FINANCE

Right click on the Specific Profile **MONEY** and then click on **New - Group Authority**. This opens the New Authorities Dialog.



Entity type: Group
Entity name: FINANCE
Object type: Topic
Profile name: MONEY
Queue manager name: WMQ7

- [ ] Change
- [ ] Clear
- [ ] Delete
- [ ] Display
- [ ] Ctrl

- [ ] Alternate user authority
- [✓] Publish
- [ ] Subscribe
- [ ] Resume

Figure 9-19 Setting new Authorities for group FINANCE

A range of authorities can be enabled for the new group **FINANCE** as shown above. Click on OK and this creates a new group authority with the authority preferences you selected.

9.1.9 Setting up Topic Security using setmqaut

The **`setmqaut`** command is used to change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups. The setmqaut command can be used both to grant an authorization, that is, give a principal or user group permission to perform an operation, and to revoke an authorization, that is, remove the permission to perform an operation. You must specify the principals and user groups to which the authorizations apply, the queue manager, object type, and the profile name identifying the object or objects. The below mentioned example demonstrates the use of setmqaut with the topic type. Also, it showcases the use of the three new kinds of authorities used: pub, sub and resume.

Example 9-3 Using setmqaut

```
To let the users group subscribe to topic DELI on Queue Manager WMQ7:

setmqaut -m WMQ7 -n DELI-t topic -g users +sub
```

```
And to allow the journalist group to publish:

setmqaut -m WMQ7 -n DELI -t topic -g journalist +pub
```

Note: Using -pub/-sub means that the authority to do publish or subscribe on that particular topic is now cleared. Also say, if a user/group has a +sub for a parent admin node, and a -sub for the child of this node, the -sub is ineffective. This holds true because the security attributes are delegated from the parent to the child admin nodes.

For a comprehensive list of options that can be used with the **`setmqaut`** command, refer to the WebSphere MQ v7.0 Infocenter, under the System Administration Guide. Then navigate to WebSphere MQ Control Commands, go to The Control Commands and click on setmqaut.

9.1.10 Mapping Queue Aliases to a Topic Object using MQSC

The queue alias can be defined with a name matching the name of the original queue to create an alias queue. However this queue alias object can be now mapped to a topic object. Running the command as listed in Example 9-4, results in all of the 'common queue attributes' being copied from the original queue definition to the alias queue definition which then refers to a topic object because the TARGET attribute in the alias queue is set to a subscription point in the topic tree

Example 9-4 Mapping Queue Aliases to a Topic Object

```
DEFINE QALIAS(q_name1) TARGTYPE(TOPIC) TARGET(topic_name)
```

Note: The existing TARGQ attribute in WebSphere MQ v6.0, currently defined as the name of the queue that the alias queue resolves to, is renamed to TARGET and generalized to allow either a queue or an administrative topic name to be specified. The attribute name TARGQ is retained as a synonym to TARGET for backward compatibility. A new MQSC attribute, TARGTYPE, is added to the MQSC DEFINE QALIAS command to define the type of object specified in the TARGET attribute.

9.2 Managing Subscriptions

WebSphere MQ v7.0 provides for management of Publications and Subscriptions through MQ Explorer and MQSC.

9.2.1 Using MQ Explorer

Expand the Queue Manager folder on the Navigator View in WebSphere MQ v7.0 explorer, and then right click on **Subscriptions**.

There are two options on the right click menu New - Subscriptions and Status.

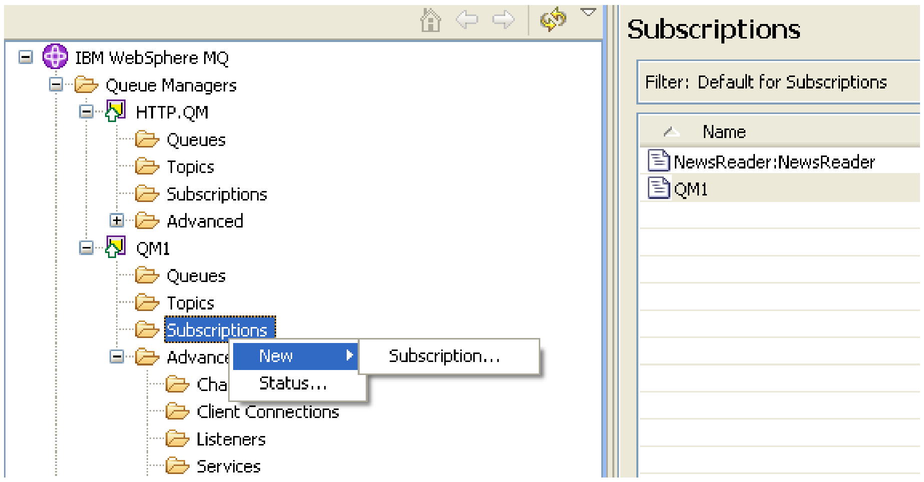
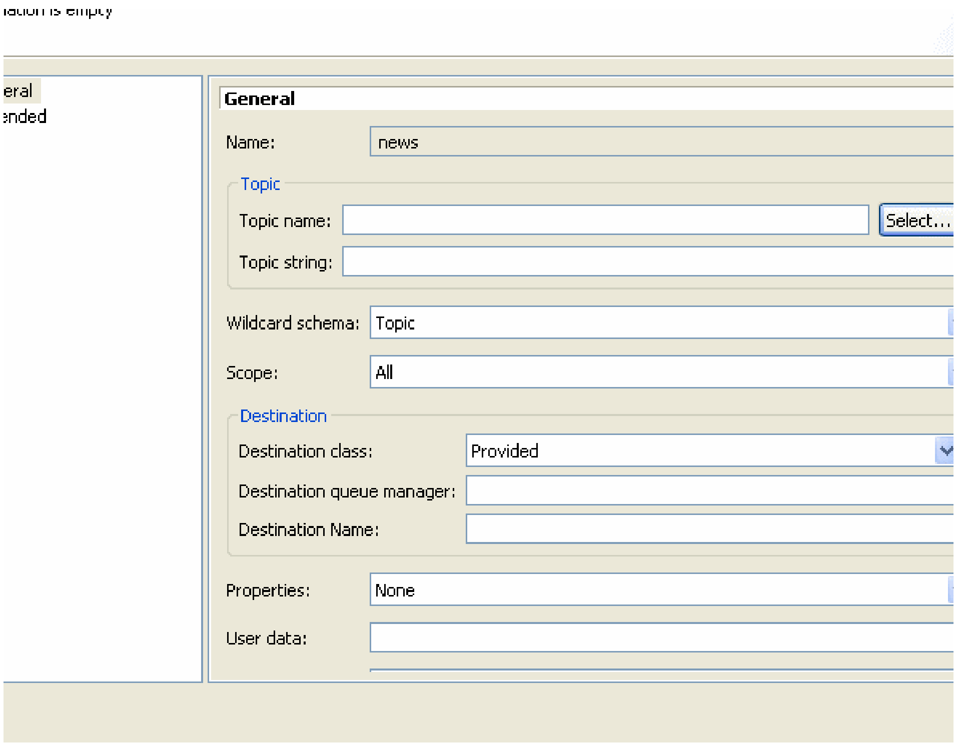


Figure 9-20 Subscriptions - Right Click Menu

Clicking on Subscription, takes you to the New Subscription Dialog Box. Enter a name for the Subscription and then click **Next**.



ation is empty

eral
ended

General

Name: news

Topic
Topic name: Select...
Topic string:

Wildcard schema: Topic

Scope: All

Destination
Destination class: Provided
Destination queue manager:
Destination Name:

Properties: None

User data:

Figure 9-21 Create Subscription

The Topic Name and the Topic String refer to the topic object name and the corresponding hierarchy respectively (`for example, matt/reatail/cat`).

The Wildcard schema can be set as Topic which means the `hash (#) or the plus (+)` or to Character, which means the asterisk (*) and the question mark (?).

Scope refers to the Subscription Scope (SUBSCOPE) which can be either set to Queue Manager or ALL, as explained earlier.

The Destination class can be set to Managed or Provided depending upon the kind of subscription that needs to be setup, which is Managed or Non Managed. If the subscription type is Non-managed, then a destination where the messages are expected to be delivered needs to be defined. So, the destination's name needs to be entered in the Destination field and the queue manager on which the destination is defined needs to be entered in the Destination Queue Manager field. In order to display the status of the subscriptions, right click on the Subscription name in the general MQ Explorer view, to open the menu as displayed in the following figure.

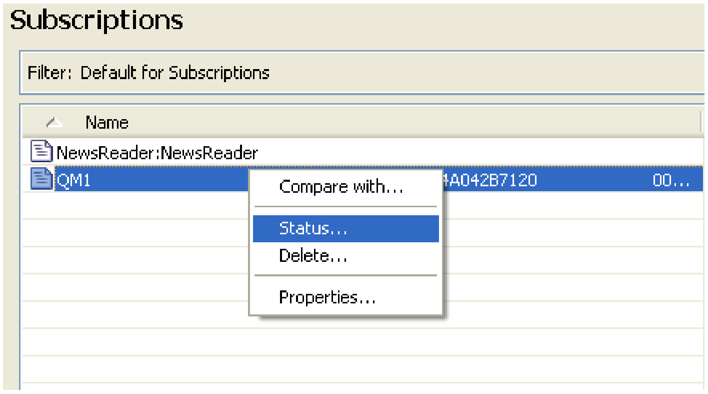
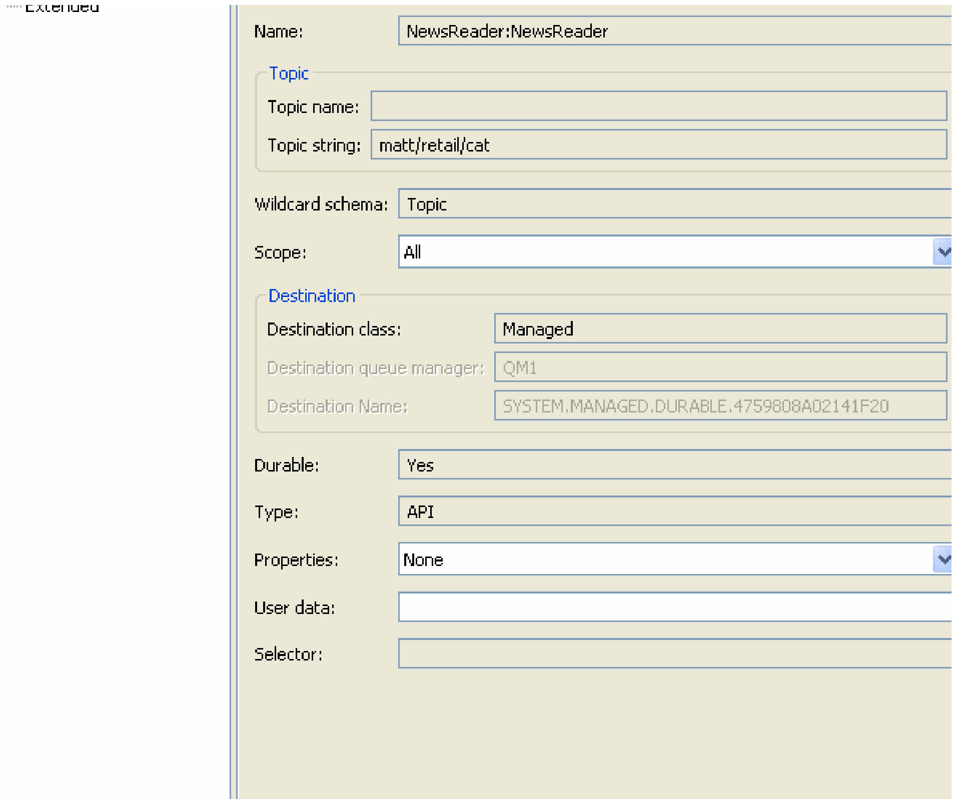


Figure 9-22 Displaying Subscription Status.

Clicking on Compare With helps you compare this subscription with other subscriptions. Clicking on Status opens the following dialog box:



Extended

Name: NewsReader:NewsReader

Topic

Topic name:

Topic string: matt/retail/cat

Wildcard schema: Topic

Scope: All

Destination

Destination class: Managed

Destination queue manager: QM1

Destination Name: SYSTEM.MANAGED.DURABLE.4759808A02141F20

Durable: Yes

Type: API

Properties: None

User data:

Selector:

Figure 9-23 Changing Subscription Properties

This view provides information about all subscriptions defined in the system. The view involves the value of all properties for existing subscriptions.

9.2.2 Using MQSC

WebSphere MQ v7.0 introduces new MQSC commands for managing subscriptions.

Define Subscription

The DEFINE SUB command allows for the administrative creation of a subscription. Example usage of this command is specified below.

Example 9-5 Usage of the DEFINE SUB Command

```
DEFINE SUB (EXSUB) TOPICOBJ(MATT.RETAIL.CAT) TOPICSTR(matt/retail/cat)
```

The TOPICOBJ and TOPICSTR can be used interchangeably to refer to a topic for which the subscription is being defined. In WebSphere MQ v7.0, there is a way to specify a destination where the subscribing application would want the messages delivered. This option is referred to as **Non-Managed Subscriptions**.This is possible by specifying the DEST, which stands for the name of the destination and the DESTQMGR refers to the name of the Queue Manager on which the destination is defined. In this case, you need to provide the option DESTCLAS(PROVIDED). There can be even a way to do **Managed Subscriptions,** which is a way of saying that the subscribing application needs the messages to be delivered to a WebSphere MQ v7.0 system destination. This can be done by specifying the option DESTCLAS(MANAGED).

Chapter 4, “Publish/Subscribe integration” on page 45 discusses under the section Distributed Publish/Subscribe the concept of Scope of Publications and Subscriptions. You can use the option SUBSCOPE to define the scope of subscriptions with the **DEFINE SUB** command. Valid parameters that can be passed to this option are QMGR which means that matching publications are delivered to subscribers which are local to the Queue Manager. ALL means that publications are delivered to both local as well as remote subscribers in a distributed publish/subscribe environment.

A way to reset the subscription to its initial state is to use the **CLEAR SUB** command.

Example 9-6 Usage of the CLEAR SUB Command

```
CLEAR SUB (EXSUB)
```

Existing subscriptions can be altered using the ALTER SUB command.

Example 9-7 Usage of the ALTER SUB Command

```
ALTER SUB (EXSUB) DEST(MATT.RETAIL.CAT.QUEUE) DESTCLAS(PROVIDED)
DESTCORL(1)
```

The Alter Sub command accepts a variety of parameter options, information about which can be found on the WebSphere MQ v7.0 Infocenter. The DESTCORL option refers to the the Corelation ID used for messages published to this subscription.

Display Publish Subscribe Status

The **DISPLAY PUBSUB** command can be used to display the publish/subscribe status information about a Queue Manager.

Example 9-8 Usage of the DISPLAY PUBSUB Command

```
DISPLAY PUBSUB <QMGR> TYPE <OPTION>
```

The TYPE parameter allows for specifying a preference in the type of Publish Subscribe information requested. It can take the following values:

ALL: Display the publish/subscribe status for this queue manager and for parent and child hierarchical connections.

CHILD: Display the publish/subscribe status for child connections.

LOCAL: Display the publish/subscribe status for this queue manager.

PARENT: Display the publish/subscribe status for the parent connection.

This command returns a set of options. The important ones are discussed below and for information on all other options, refer the WebSphere MQ v7.0 Infocenter.

The Status Information returned by this command should be interpreted as under:

If the TYPE entered was LOCAL, the following values can be returned:

ACTIVE**:** The publish/subscribe engine and the queued publish/subscribe interface are running. It is therefore possible to publish or subscribe using the application programming interface and the queues that are monitored by the queued publish/subscribe interface.

COMPAT: The publish/subscribe engine is running. It is therefore possible to publish or subscribe by using the application programming interface. The queued publish/subscribe interface is not running. Therefore, any message that is put to the queues that are monitored by the queued publish/subscribe interface is not acted upon by WebSphere MQ v7.0.

ERROR: The publish/subscribe engine has failed. Check your error logs to determine the reason for the failure.

INACTIVE: The publish/subscribe engine and the queued publish/subscribe interface are not running. It is therefore not possible to publish or subscribe using the application programming interface. Any publish/subscribe messages that are put to the queues that are monitored by the queued publish/subscribe interface is not acted upon by Websphere MQ v7.0.

STARTING:The publish/subscribe engine is initializing and is not yet operational.

STOPPING: The publish/subscribe engine is stopping.

If the `TYPE` entered was `CHILD`, the following values can be returned:

ACTIVE: The connection with the child queue manager is active.

ERROR**:** This queue manager is unable to initialize a connection with the child queue manager because of a configuration error. Possible causes include: a) Transmit queue is not defined or b) Transmit queue put is disabled.

STARTING: Another queue manager is attempting to request that this queue manager become its parent.

STOPPING: The queue manager is disconnecting.

If the `TYPE` entered was `PARENT`, the following values can be returned:

ACTIVE: The connection with the parent queue manager is active.

ERROR: This queue manager is unable to initialize a connection with the parent queue manager because of a configuration error. Possible causes include : a) Transmit queue is not defined. b)Transmit queue put is disabled.

REFUSED:The connection has been refused by the parent queue manager. This might be caused by the following:

The parent queue manager already has a child queue manager with the same name as this queue manager.

The parent queue manager has used the command **`RESET QMGR TYPE(PUBSUB) CHILD`** to remove this queue manager as one of its children.

STARTING: The queue manager is attempting to request that another queue manager become its parent.

STOPPING: The queue manager is disconnecting from its parent.

Display Subscriber Status

The **DISPLAY SBSTATUS** command can be used to display the status of a given subscription. The syntax for using this command can be specified as:

Example 9-9 Usage of DISPLAY SBSTATUS Command

```
DISPLAY SBSTATUS <NAME> WHERE <KEYWORD> <OPERATOR> <VALUE>
```

Name refers to the name of the subscription for which status information needs to be displayed. The Keyword refers to any attribute that you can use with the `DEFINE SUB` Command, except for the `CMDSCOPE` option. Valid operators that can be used are Less than (LT), Greater than (GT), Equal to (EQ), Not Equal to (NE), Less than or equal to (LE), Greater or equal to (GE), Match a generic string (LK), Does not match a generic string (NL)

Two types of values can be used in this command. An explicit value is a valid value that can be used for the corresponding attribute that is provided in the `Keyword`. A generic value is a character string with an asterisk at the end. for example ABC*. If the operator is LK, all items where the attribute value begins with the string (ABC in the example) are listed. If the operator is NL, all items where the attribute value does not begin with the string are listed.

Note: Additional parameter options that can be used with the commands discussed above are explained in the WebSphere MQ v7.0 Infocenter.

10

z/OS enhancements

This chapter describes the enhancements to WebSphere MQ V7.0 for the z/OS operating system. They should be considered before installing or migrating to WebSphere MQ V7.0 on z/OS.

The sections in this chapter cover each of the enhancements:

- “Publish/Subscribe for z/OS”
- “RACF mixed case classes and profiles”
- “Using WebSphere MQ Explorer without CAF”
- “WebSphere MQ for z/OS listener”
- “CICS OTE”

For additional information refer to the following WebSphere MQ manuals:

Manual *WebSphere MQ Systems Administration Guide V7.0*, XXXXXX
Manual *WebSphere MQ z/OS Concepts and Planning Guide V7.0*, XXXXXX
Manual *WebSphere MQ z/OS Systems Setup Guide V7.0*, XXXXXX
Manual WebSphere MQ Publish/subscribe Users Guide V7.0, XXXXXX
Manual *WebSphere MQ Security V7.0*, XXXXXX

These manuals can be viewed online in the IBM Information center, available at:

https://wmqi-id.hursley.ibm.com/wmq/help/index.jsp

10.1 Publish/Subscribe for z/OS

WebSphere MQ V7.0 for z/OS now provides a native Publish/Subscribe (Pub/Sub) feature, exactly as implemented on the distributed platforms in this version.

The Pub/Sub feature was first introduced to WebSphere MQ in V5.3 for distributed platforms. It was previously only available on the z/OS platform in WebSphere Message Broker.

For detailed information about Pub/Sub in V7.0 refer to Chapter 4., “Publish/Subscribe integration” on page 45.

10.2 RACF mixed case classes and profiles

WebSphere MQ V7.0 uses five new RACF classes. They allow profiles to be defined to protect access to mixed case object names and administration functions:

Table 10-1 New RACF classes

New RACF Class name	Use
MXADMIN	Adminstration
MXQUEUE	Queues
MXPROC	Processes
MXNLIST	Name Lists
MXTOPIC	Publish/Subscribe Topics

Mixed case object names can be protected by defining:

- Mixed case profiles in the appropriate mixed case RACF class, or,
- Generic upper case profiles in the appropriate upper case RACF class.

The new MQXTOPIC class does not have an equivalent upper case class.

10.2.1 New queue manager parameter SCYCASE

The new queue manager parameter SCYCASE (Security Case) determines whether UPPER or MIXED case profiles are used by the queue manager:

- UPPER: Only upper case profiles are used.
- MIXED: Mixed case profiles are used, in addition to upper case profiles.

Changes to this parameter are only effective after a successful **`REFRESH SECURITY(*) TYPE(CLASSES)`** command or after the queue manager has been recycled. This parameter is only valid on z/OS.

The **`DISPLAY QMGR SCYCASE`** command displays the current setting of the SCYCASE parameter for the queue manager. This parameter is only valid on z/OS.

Note to Reviewer:
It still needs to be confirmed that WMQ Explorer displays SCYCASE for z/OS queue managers, as this was unable to be done during the Nov-Dec 2007 residency. Appropriate information needs to be added to this section.

10.2.2 Upper case only profiles

There are some profiles, or parts of profiles, that are required by WebSphere MQ to be in upper case. These are:

- All high-level qualifiers (HLQ) for subsystem identifiers and queue sharing group identifiers.
- Profiles for SYSTEM objects, including the default objects.
- The MQCMDS class, as all command profiles are upper case only.
- The MQCONN class, as all connection profiles are upper case only.
- Switch profiles.
- RESLEVEL profiles.
- The command type in command resource profiles. For example, hlq.QUEUE.queuename.
- Dynamic queue profiles used by WebSphere MQ administration tools. For example, hlq.CSQOREXX.*, hlq.CSQUTIL.*, and CSQXCMD.*
- The 'CONTEXT' part of the hlq.CONTEXT.resourcename profiles.

10.2.3 Mixed case profiles

The following RACF commands illustrate the advantage of using mixed case profiles when mixed case object names exist in the queue manager.

The mixed case class MXQUEUE can be used to explicitly protect a queue called PAYROLL.Dept1 on a queue manager called QMHQ:

```
RDEFINE MXQUEUE QMHQ.PAYROLL.Dept1
```

The upper case class MQQUEUE can only be used to protect this queue by referring to the high level qualifier in the queue name:

```
RDEFINE MQQUEUE QMHQ.PAYROLL.**
```

This profile also protects all other queues which begin with PAYROLL, which may be an undesirable situation for the queue manager security.

10.2.4 Refreshing mixed case profiles

The REFRESH SECURITY command has been updated to allow the new RACF classes to be refreshed in a queue manager.

For example, the following command deletes all the profiles in the MXQUEUE class which are held in storage by the queue manager:

```
REFRESH SECURITY(MXQUEUE)
```

Profiles are then loaded into queue manager storage from RACF as they are needed to perform security validations.

10.3 Using WebSphere MQ Explorer without CAF

MQ Explorer can be used to remotely administer and monitor MQ objects, including topics and other Publish/Subscribe facilities.

WebSphere MQ V7.0 for z/OS introduces a limited capability to allow MQ Explorer and other programs such as SupportPac M071 to administer z/OS queue managers at this version without purchasing a license for the Client Attach Facility (CAF). A license still needs to be purchased to allow any other type of WebSphere MQ Client application to connect to the queue manager.

This makes available the great benefits of using the features and graphical user interface of MQ Explorer to administer z/OS queue managers which did not previously have this license.

Up to five client programs can connect to each z/OS queue manager via the SYSTEM.ADMIN.SVRCONN channel. There is a sample definition of this SVRCONN type channel in the CSQ4INSG member. This must be copied to a member which is concatenated to CSQINP2 DD of the MSTR address space.

The new MAXINST parameter must be added to the definition with a numeric argument in the range of 1 to 5, to limit the maximum number of concurrently running channel instances from MQ Explorer users.

Example 10-1 The following MQSC command specifies a limit of 5 users:

```
DEFINE CHANNEL( 'SYSTEM.ADMIN.SVRCONN' ) +
CHLTYPE( SVRCONN ) +
QSGDISP( QMGR ) +
* Server-connection channel attributes
DESCR( 'System-command client channel' ) +
TRPTYPE( TCP ) +
MCAUSER( ' ' ) +
SCYEXIT( ' ' ) SCYDATA( ' ' ) +
MSGEXIT( ' ' ) MSGDATA( ' ' ) +
SENDEXIT( ' ' ) SENDDATA( ' ' ) +
RCVEXIT( ' ' ) RCVDATA( ' ' ) +
PUTAUT( DEF ) +
DISCINT( 0 ) KAINT( AUTO ) +
COMPHDR( NONE ) COMPMSG( NONE ) +
MONCHL( QMGR ) +
SSLCIPH( ' ' ) +
SSLPEER( ' ' ) +
SSLCAUTH( REQUIRED ) +
MAXMSGL( 4194304 ) +
MAXINST( 5 )
```

It is recommended that security should be imposed on the SYSTEM.ADMIN.SVRCONN channel by using Secure Socket Layer (SSL) or a Security Exit (such as IBM SupportPac MS0R) to restrict access to authorized administrators. Security enhancements for remote administration using MQ Explorer are covered in the section 8.1.5, “Security” on page 165. This contains all steps required to set up security on the Explorer side.

MS0R SupportPac can be downloaded from the following link:

http://www.ibm.com/support/docview.wss?rs=1083&uid=swg24015658

10.4 WebSphere MQ for z/OS listener

In WebSphere MQ for z/OS, listening for connections on TCP/IP and LU6.2 protocols is performed within the channel initiator (CHIN) address space of a queue manager. The CHIN also runs all message channel agents (MCAs) for the queue manager, whether they are a distributed message channel, cluster message channel or client connection channel.

A listener is usually started and stopped on a z/OS queue manager using a command interface on the z/OS system.

WebSphere MQ Explorer V7.0 now supports the remote starting and stopping of the z/OS listener. This feature was not provided in prior versions of the MQ Explorer.

It is now also possible to define new listeners on z/OS queue managers using MQ Explorer. The procedure is as follows:

1. Right click on the **Listeners** folder for the z/OS queue manager that is to contain the new listener, then click **New** and select the type of listener to be defined (TCP or LU6.2). The **New Listener** dialog will then open.
2. Make all the necessary changes to the attributes now, because z/OS listener attributes are configurable only when they are initiated, and cannot be altered later.
3. Click **Finish** to start the new listener.
4. The new listener is started and displayed in the Content view.

10.5 CICS OTE

The CICS Open Transaction Environment (OTE) allows transactions to run under their own Task Control Blocks (TCBs) rather than all running on the Quasi-Reentrant (QR) TCB which is normally used.

To run an Open TCB, code must be thread-safe. This means it is not dependant on the serialization that the QR TCB provides, but uses proper serialization techniques when updating shared resources.

CICS automatically provides the necessary TCB switches when jumping between thread-safe and non thread-safe code. The MQ Task Related User Exit (CSQCTRUE) used by all MQ API calls is not currently thread-safe so all WebSphere MQ calls from an Open TCB switch to the QR TCB, and then from CSQCTRUE to one of the eight RMI TCBs to actually perform the real work. Return from RMI TCB is also via the QR TCBs.

Making CSQCTRUE thread-safe eliminates the TCB switches to and from the QR TCB, and also the TCB switches to and from the RMI TCBs.

The queue manager CTHREAD parameter is ignored by CICS OTE, since it does not provide a useful function in controlling resource usage by the feature.

The benefits are:

- No external change for applications.
- Exits (data conversion, API crossing) need to be thread-safe.
- If not declared thread-safe, WebSphere MQ reverts to previous behavior.
- More efficient use of TCBs, especially when mixing calls to WebSphere MQ and IBM DB2.

11 Installation and migration

Note to Reviewer: There is uncertainty about the description of recoverability in this chapter and how pub/sub migration will function. We believe that there are contradictions in the DCR's about how the recovery will work (especialy PSMODE description), or it is not clear to us. We were not able to test the recoverability as the functionality was not available in ALPHA IT5 code. This chapter must be revisited (especially the pub/sub migration description) to ensure what is described here is the way the Beta code and GA code actually function. We would very much welcome any input from the developent team as to how the pub/sub migration must actually performed.
Bill Sapolyo & Glenn Baddeley, SAM725 residents, Hursley, Nov-Dec 2007.

This chapter provides guidance on some of the considerations which should be made prior to installing WebSphere MQ V7.0 or performing a migration from previous versions. It is not possible to cover all migration issues in the scope of this book due to the many complex factors related to platform environments and how applications use WebSphere MQ.

There are no significant changes to the installation process for WebSphere MQ V7.0 when compared with previous versions, however hardware and software prerequisites have changed and supported operating system versions have moved forward.

The biggest impact is to current users of Publish/Subscribe with WebSphere MQ. A deep understanding of the migration issues is required and detailed planning must be carried out to ensure a successful migration.

Appropriate references are made in this chapter to WebSphere MQ manuals which provide more information. These are available in the WebSphere MQ Information Center at:

`https://wmqi-id.hursley.ibm.com/wmq/help/topic/com.ibm.mq.flat.doc/mq50020_.htm`

The following topics are discussed in this chapter:

- Hardware and Software prerequisites
- Installation of WebSphere MQ V7.0
- Co-existence with previous versions
- Migration

11.1 Hardware and Software prerequisites

The Hardware and Software prerequisites for WebSphere MQ V7.0 are described in the relevant *WebSphere MQ Quick Beginnings* book for the distributed platforms, and in the *WebSphere MQ for z/OS System Setup Guide*, XXXXXX, and *WebSphere MQ for z/OS Program Directory*, XXXXXX, for the z/OS platform.

Detailed migration information is found in *Migrating to WebSphere MQ V7.0, XXXXXX*. This manual is strongly recommended reading.

National Language support on AIX

WebSphere MQ V7.0 requires an AIX® operating system at level 5.3 or higher. From this level of operating system onwards the IBM-850 code pages are no longer supported.

If any of the IBM-850 code pages are installed these must be removed manually before installing WebSphere MQ V7.0 and before upgrading to the AIX 5.3 operating system.

Microsoft Vista support with WebSphere MQ

WebSphere MQ V7.0 introduces support for the Microsoft Vista operating system.

If migrating from previous operating systems, such as Microsoft Windows XP, the existing WebSphere MQ must first be upgraded to WebSphere MQ V7.0 before upgrading the operating system. Prior versions of WebSphere MQ do not provide support for the Microsoft Vista operating system.

11.2 Installation of WebSphere MQ V7.0

There are no changes to the installation process in WebSphere MQ V7.0. The recommendation is to review the standard installation documentation specific to the platform on which WebSphere MQ V7.0 is being installed.

The installation of WebSphere MQ and the configuration of new queue managers is described in the relevant *WebSphere MQ Quick Beginnings* book for the distributed platforms, and for z/OS in the *WebSphere MQ for z/OS System Setup Guide, XXXXXX*, *WebSphere MQ for z/OS Concepts and Planning Guide , XXXXXX*, and the *WebSphere MQ for z/OS Program Directory*.

11.3 Co-existence with previous versions

On distributed platforms it is not possible to install and run WebSphere MQ V7.0 to co-exist with an earlier version of WebSphere MQ on the same server. Installing WebSphere MQ V7.0 always upgrades any earlier version of WebSphere MQ which has been previously installed on the server.

Co-existence is only supported on the z/OS platform, where each queue manager subsystem can have its own set of product datasets running at different versions of WebSphere MQ.

For z/OS, ensure that the WebSphere MQ V7.0 early code is installed and the Coexistence PTFs are applied to the existing queue managers, before performing the upgrade. Do this activity well ahead of the upgrade and ensure the current environment is stable with the PTFs before proceeding to the upgrade.

WebSphere MQ V7.0 interoperates within a network of existing queue managers running on previous versions of WebSphere MQ, using clients, distributed queueing and MQ clusters.

For managing coexistence of message properties between MQ V7.0 and other versions, refer to PROPCTL attribute on channels and queues, as described in the relevant *WebSphere MQ Quick Beginnings* manual.

11.4 Migration

The migration of every possible combination of a WebSphere MQ version, supported platform, infrastructure software and associated applications is not within the scope of this book. This section highlights the main areas of impact when migrating to WebSphere MQ V7.0 from previous releases and gives some guidance on actions which can be taken to ensure a smooth migration.

Prior to the installation or migration of any software it is recommended that a full system backup is performed to ensure that the system can be reverted to its original working environment in the event that unresolvable problems are encountered during the migration.

11.4.1 General Migration considerations

The following lists some of the general issues to consider when planning a migration.

- Detailed migration information is provided in the manual *Migrating to WebSphere MQ V7.0, XXXXXX.*
- The *WebSphere MQ Quick Beginnings Guide* for specific distributed platforms and the *WebSphere MQ z/OS System Setup Guide XXXXXX*, should be consulted.
- Review the latest README files which are shipped with the product. They many contain information which is not included in the manuals.
- Develop a backup plan to capture all current data of WebSphere MQ. Queue manager attributes, object definitions and messages are saved while the queue manager is running and without any active applications or channels. The WebSphere MQ file systems are saved while the queue manager is not running to ensure the backup has a consistent state.
- This is also a good time to check through all existing queue managers to see whether there are queues and other objects which are no longer needed, and whether there are any queue managers that are no longer required. Some queue managers may need to be kept at an earlier version due to application dependencies and a lack of prerequisite software. They can be still be administered from a migrated system.
- Document and review details about the existing system topology, including the names of the queue managers and their queues, channels, and how they relate to applications and other queue managers.
- If inetd (Unix network daemon) or a manually started **`runmqlsr`** is being used as the MQ TCP/IP port listener, the migration is a good opportunity to change to using a LISTENER object in the queue manager.
- Conduct thorough functionality testing after migrating a development or test environment before migrating a production environment.
- If it is not possible to shut down a queue manager to be migrated due to availability requirements, a different migration approach may need to be considered. The migration can be performed using the following general steps:
  - Copy all resources from the server concerned to another server.
  - Perform a migration on the duplicate server.
  - Switch over the workload to the queue manager on the duplicate server at a convenient quiet time when there are no application messages queued. This may involve altering channels or DNS entries.

11.4.2 Queue manager Migration

Before migrating to WebSphere MQ V7.0, ensure that all prerequisites are satisfied and current environment is backed up.

Ensure that applications are thoroughly tested against the product before deploying into production. It is recommended that the migration be completed in two phases.

- Perform the migration without introducing any application changes or to start using any of the new features of WebSphere MQ V7.0.
- Once there is satisfaction that the product is functioning as expected within the existing business environment and applications, introduce the new and enhanced features provided in this release, and test the applications thoroughly.

11.4.3 Migration considerations:

The following is an example of a process which could be used to migrate a queue manager to WebSphere MQ V7.0 from a previous version:

1. Stop all channels. Ensure that all channels are also stopped on remote queue managers which connect to this queue manager.
2. Stop the listener(s).
3. Stop all applications that can connect to the queue manager.
4. Stop system management products such as monitors and automation tools that can connect to the queue manager.
5. Unload all messages currently queued to a file.
6. Save all queue manager attributes and object definitions.
7. Save all the security profile settings. These are held in OAM on distributed platforms and RACF or ACF2 on z/OS.
8. Shutdown the WebSphere MQ queue manager(s) and associated services, such as dead letter queue handlers and trigger monitors.
9. Backup the current environment, ensuring that there are two copies on separately located media. If time and resources are available, it can be worth while to ensure the integrity of the product backup by testing restoration on a separate server.
10. Uninstall the existing version of WebSphere MQ. For z/OS, ensure the pagesets and logs remain, but remove or rename the product libraries.
11. Perform the installation steps as described in the WebSphere MQ V7.0 product documentation for the specific platform type.

12. Start the WebSphere MQ queue manager(s) and wait for it to complete the automatic migration of the MQ file system structure to the format used by the new version.
13. If the LISTENER object is not being used to to start the MQ TCP/IP port listener, start the listener process **`runmqlsr`** or enable the port in inetd.
14. Start channels which were stopped before the upgrade.

11.4.4 Fallback considerations

In the event that the migration needs to be backed out to the previous version, the following steps are recommended. These apply to distributed platforms only. For z/OS, consult the specific product documentation for more details.

1. Stop all channels. Ensure that all channels are also stopped on remote queue managers which connect to this queue manager.
2. Stop the listener(s).
3. Stop all applications that can connect to the queue manager.
4. Stop system management products.
5. Unload all messages currently queued to a file.
6. Save all queue manager attributes and object definitions.
7. Shutdown the WebSphere MQ queue manager(s) and associated services, such as dead letter queue handlers and trigger monitors.
8. Uninstall the WebSphere MQ V7.0 product.
9. Using the backup taken prior to the upgrade, restore the previous version of the WebSphere MQ product.

Note: Any changes to queue manager and object defintions introduced in WebSphere MQ V7.0 are not available.

Note: In the event of a fallback on z/OS the product libraries would be renamed. Do not remove or delete the existing logs, archives and pag-esets.

10. Start the queue manager, ensuring that all channels and listeners are stopped.
11. Remove any existing messages from the queues.
12. Load messages from the file generated in step 5.

13. Perform verification tests.
14. Start the listener and channels and so on.
15. Continue to monitor the system health to ensure the prior version of WebSphere MQ is functioning correctly.

11.4.5 Publish/Subscribe engine

WebSphere MQ V7.0 provides a new Pub/Sub engine which is integrated into the queue manager. Pub/Sub was available with WebSphere MQ V6.0 on distributed platforms using a separate broker process which was associated with each queue manager. It was also available on z/OS by using WebSphere Message Broker / Event Broker. The Pub/Sub engines which worked with WebSphere MQ V6.0 (and WebSphere MQ V5.3 with FixPack 8 and onwards) are now deprecated, however applications written for them can work with WebSphere MQ V7.0 without modification.

To allow allow compatibility for applications which used the older Pub/Sub engines a queued Pub/Sub interface is provided in WebSphere MQ V7.0. The operation of this interface is controlled via a new queue manager attribute called PSMODE. The older MQ Pub/Sub applications can continue running without modification until such time that the PSMODE is set to permanently disable the queued Pub/Sub interface.

WebSphere MQ V7.0 Queued Pub/Sub interface start-up

Unlike the old V6.0 Pub/Sub engine, there are no commands which control the queued Pub/Sub interface. It is now controlled by setting a new a queue manager attribute PSMODE.

PSMODE has three settings:

- DISABLED, all Pub/Sub functionality disabled.
- COMPAT, the new V7.0 Pub/Sub engine enabled but the queued Pub/Sub interface disabled.
- ENABLED, both the new V7.0 Pub/Sub engine and the queued Pub/Sub interface are enabled.

The initial setting of the queue manager attribute is ENABLED for newly created queue managers and COMPAT for existing queue managers, however the COMPAT state may be changed during V6.0 to V7.0 Pub/Sub migration.

Once migration has completed successfully the PSMODE attribute is set to ENABLED.

The **dltmqbrk** control command is used to delete the V6.0 broker and its associated state. If **dltmqbrk** is issued once more it produces a warning that no broker state is present and no action is taken.

After a successful migration, the broker records that it has completed the migration. If **strmqbrk** is then run on the queue manager, it checks for this value and warns the user that migration is complete. If a forced migration is requested (with the **-f** option) then a complete migration is then performed even if migration has already been successfully completed.

The default value of PSMODE is DISABLED for migrated queue managers. PSMODE must be is set to DISABLED when using V7.0 queue manager with WebSphere Message Broker V6.0 in order to disable the WebSphere MQ V6.0 Pub/Sub queued interface.

IBM encourages customers to migrate WebSphere MQ V6.0 Pub/Sub objects and configurations to the new WebSphere MQ V7.0 Pub/Sub engine as soon as practical.

11.4.6 Applications Migration considerations

There are no migration processes or changes required to application programs for migration or installation of WebSphere MQ V7.0. Existing applications will continue to function in their current state.

The WebSphere MQ V6.0 Pub/Sub engine can be replaced with a daemon process in WebSphere MQ V7.0 which supports existing WebSphere MQ V6.0 applications.The daemon process services old style applications by mapping their queued Pub/Sub interface into the new WebSphere MQ V7.0 API and native Pub/Sub engine.

The daemon is started like the WebSphere MQ V6.0 broker, by issuing the WebSphere MQ V6.0 control command **strmqbrk** or from the WebSphere MQ V6.0 service **runmqbrk**.

RFH1, RFH2 and PCF format message headers are supported.

Subscribers receive the appropriate headers for their subscription. For example, a subscription that used an RFH to subscribe will receive RFH publications, whatever format the publisher originally used.

WebSphere MQ V6.0 and WebSphere Message Broker V6.0 style wildcards are both supported via subscription options.

As WebSphere MQ V7.0 uses synchronous calls to its native Pub/Sub engine, any response messages that are required to old style Pub/Sub applications can be built up by the daemon process and put to the appropriate queue as specified by the ReplyToQ and ReplyToQMgr fields in the Message Descriptor (MQMD).

Content-based filtering and transformations continue to be managed by WebSphere Message Broker.

The command **migmbbrk** is used to migrate from WebSphere Message Broker / WebSphere Event Broker Publish/Subscribe. This supports WebSphere Message Broker V6.0 Delete Publication, Publish and Subscription.

WebSphere MQ V6.0 StreamName and Topic parameters are used together to produce a WebSphere MQ V7.0 topic object and topic string. This provides support for Retained publications and heirarchical inter-broker relationships.

11.4.7 WebSphere MQ Clients

Client channel definitions which are written to a Client Channel Definition Table (CCDT) file contain an inherent structure version for each channel. The version of the structure dictates which versions of WebSphere MQ can read the definition. It is not possible to define a client channel in a CCDT using WebSphere MQ V7.0 and have it read by a WebSphere MQ V6.0 client or an earlier version client.

Client Channel Definition Table (CCDT) files are not migrated when upgrading to WebSphere MQ V7.0. Nor is the entire file updated when updating just one channel with in the CCDT.

Client channels in CCDT files should be defined and maintained on a queue manager which is at the same version or earlier than the versions of all the clients that use the file. As an alternative, IBM SupportPac MO72 allows earlier version CCDT files to be maintained on any system without requiring an earlier version queue manager. The SupportPac is available for download at:

http://www.ibm.com/support/docview.wss?uid=swg24007769

Part 3

Scenario

This part contains the scenario which demonstrates how the new features and enhancements work and how to use them. The sample programs and scripts used for this scenario are available as text in Appendix A, “Scenario preparation scripts” on page 337 and for download by following the instructions in Appendix B, “Additional material” on page 343. This part consists of:

12

Scenario overview

Author Comment: Section **12.3.3 and 12.3.4** are still to be written.

Matt’s Deli is a chain of grocery stores, only a few stores right now, but growing fast and planning to become a major brand.

In the scenario that follows the new features of WebSphere MQ V7.0 are showcased while staying within the framework of a realistic business situation.

The scenario only covers part of Matt’s Deli business operations and the application programs are deliberately simplified. The intention is to illustrate usage of the new features, rather than being a guide to their best usage or to implement complete business logic.

The following topics are discussed in the scenario overview:

- “Business Environment”
- “Scenario Implementation”
- “Components”

12.1 Business Environment

Each Matt's Deli store manager has the freedom to stock products of their choice from the Matt's Deli range of products. Stores place their orders with Headquarters and they are delivered daily from the warehouse.

The newest venture is a Web Deli where customers can order products directly on Internet for home delivery.

Headquarters negotiates prices from suppliers and arranges delivery of stock to to the warehouse.

The warehouse takes orders from Headquarters and performs fulfillment of product orders to stores and Web Deli customers.

Matt's Deli has an ambitious growth plan and wants to ensure that its Information Technology systems can grow with the company. Currently all stores are in the United Kingdom (UK) but local expansion and movement into other countries are planned. In the same way, Web ordering is limited to UK delivery but there is a plan to grow volume and geographic reach. Currently a single warehouse is in operation, but additional warehouses may be needed to cope with growth and geographic spread.

12.1.1 Business flow

The business flow is illustrated in Figure 12-1 on page 241. There are five groups of participants or "actors" in the business flow. Starting in the bottom left hand corner of the figure, they are:

- Stores: Each Matt's Deli store places orders with Headquarters for items in the Matt's Deli catalog. Additionally, the stores receive news from Headquarters about promotions and other business matters.
- Internet Customers: The Web Deli can be used to place direct orders for products which are delivered from the warehouse. Internet customers can also receive news of special offers and other items of interest from Matt's Deli.
- Warehouse: Orders are received from stores and from Internet customers. The delivery fulfillment flow is not included in this scenario.
- Suppliers: Each of the suppliers to Matt's Deli must send quotes to Headquarters for their best price on products they wish to supply. Small suppliers may quote for only one or two items in the catalog, large suppliers for many or all of them. These quotes are used by Headquarters to choose the best supplier(s) of that item. The ordering process from Matt's Deli to the

suppliers is not included in this scenario. In the future Matt's Deli may use the news system to send relevant news to suppliers.

- Headquarters: This is the hub of business activity for communication between Matt's Deli systems. Orders are received from stores and from Internet. After processing they are sent to the warehouse for fulfillment. Requests for quotes may be published to suppliers, showing items for which Matt's Deli wishes to receive quotes. Quotes from suppliers are received and processed. This consists of checking if the new quote is cheaper than the existing best quote. If it is, a new retail price is published to the stores and the Internet customers. News is also generated from headquarters for viewing by interested parties at the stores and by Internet customers.

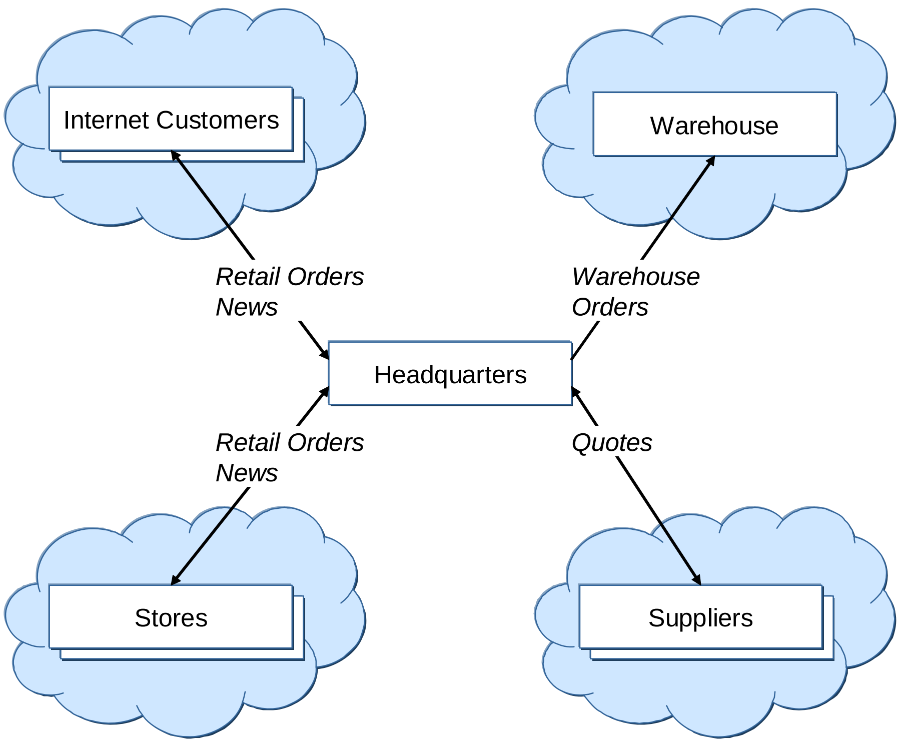


Figure 12-1 Matt's Deli High Level Data Flow

12.1.2 Choosing WebSphere MQ

In order to satisfy the business requirements described above, Matt's Deli has decided to do an implementation based on WebSphere MQ. The Publish/Subscribe paradigm was chosen because Matt's Deli needs the flexibility to grow rapidly. In particular they need to be able to:

- Communicate with a wide range of suppliers, over which Matt's Deli has no control of their systems or hardware.
- Communicate asynchronously with suppliers and stores which may have patterns of availability that differs from the Headquarters.
- Allow for rapid growth by adding stores, suppliers and warehouses without disruption to systems.

The existing point-to-point messaging features of WebSphere MQ satisfy the first two requirements easily and the integrated Publish/Subscribe capability in WebSphere MQ V7.0 is very suited to the third requirement.

12.1.3 Simplifying the scenario

In this scenario all WebSphere MQ messages have a very simple payload. This is so that small application programs can illustrate the WebSphere MQ functionality without being complicated by complex message parsing. The chosen payload for each type of message is a Comma Separated Value (CSV) message, where each field is separated from the next by a comma. All the fields in the message are simple text fields, allowing easy creation of test data.

One important field which is used in a number of messages is the *CatalogId*, which represents a product in the Matt's Deli catalog. This *CatalogId* is simply a string representing the place in the Matt's Deli product hierarchy of that product. So `Red Apples` might have a *CatalogId* of `cat/fresh/fruit/apple/red`. Using this as the product key makes it very simple to construct topic strings for Publish/Subscribe based applications.

A more realistic situation might have been to use Extensible Markup Language (XML) for the messages and to have an arbitrary product key that was used to search a database for the product hierarchy information. This would have led to more complex code and required the installation of a database as part of the illustrative scenario.

Other measures to keep the scenario code simple are not to validate message formats and field contents. It is assumed that numeric fields only contain digits and that alphanumeric fields do not contain embedded commas.

Other IT business functions of Matt’s Deli are not implemented in the scenario, such as customer billing and fulfilment of orders by the warehouse.

12.2 Scenario Implementation

Many of the new features in WebSphere MQ V7.0 are illustrated in the scenario. The intention is to show how the features might be used in a realistic situation, not to build an optimal solution to a complete set of business requirements.

The stated requirements for Matt’s Deli are satisfied in the following ways:

- Communicate with a wide range of suppliers, over which Matt’s Deli has no control of their systems or hardware.
  - Supplier applications are written in Java using JMS and also in C using the MQI.
- Communicate asynchronously with suppliers and stores which may have patterns of availability that differs from the Headquarters.
  - All applications use queued interfaces for communication, requiring only the queue manager to be available. Cooperating applications may be temporarily unavailable.
- Allow for rapid growth by adding stores, suppliers and warehouses easily and without disruption to systems.
  - Publish/Subscribe is used for supplier pricing, catalogs and news systems, allowing for simple addition of suppliers, stores and warehouses.

The scenario also illustrates other features of WebSphere MQ V7.0:

- The MQ HTTP bridge is used for web ordering, showing how messages may sent and received by a zero-footprint client (web browser) which has no WebSphere MQ code installed.
- The news applications show how the new asynchronous-put and read-ahead features might be used.
- The warehouse application shows how the new call-back feature might be used. It also uses an administrative subscription to a remote queue.
- The retail price catalog illustrates use of retained publications.

12.2.1 Application flow

The scenario application flow is illustrated in Figure 12-2 on page 244, where the five actors of the business flow are shown with more detail. The arrows indicate the flow of messages through queues and topic trees.

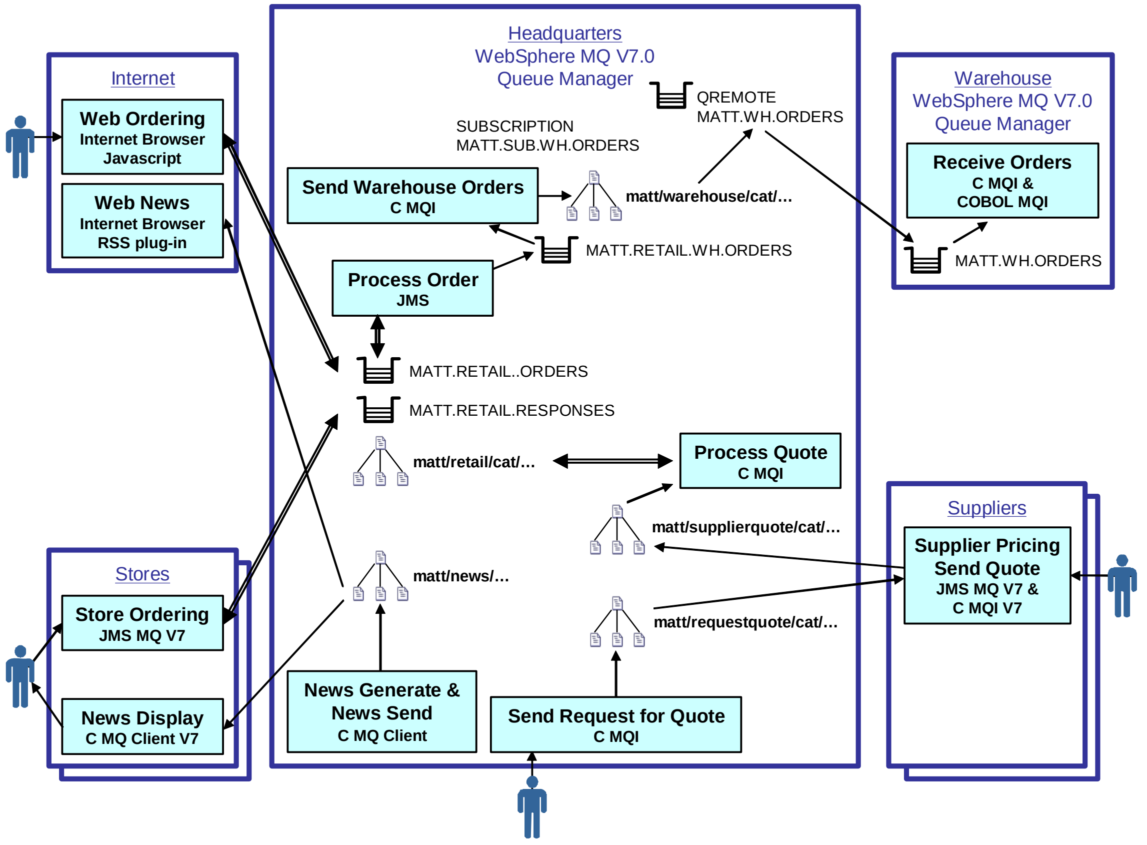


Figure 12-2 Matt's Deli application flows.

12.2.2 Infrastructure

The scenario uses the following topic objects and queue objects which are defined on the Headquarters queue manager.

The table below describes the topic object and strings used in the scenario.

Table 12-1 Topic objects and Topic strings

Topic Object Name	Topic String	Usage
MATT.TOPIC.REQUEST.QUOTE	matt/requestquote/cat/…	Matt's Deli uses this topic tree to publish requests for quotes from suppliers.
MATT.TOPIC.SUPPLIERQUOTE	matt/supplierquote/cat/...	Suppliers publish their best price for products to this topic tree.
MATT.TOPIC.RETAIL	matt/retail/cat/...	Matt's Deli publishes retail prices to this topic tree. These prices are **retained publications** so only one price can exist for each product.
MATT.TOPIC.WAREHOUSE	matt/warehouse/cat/...	Matt's Deli Headquarters publishes orders for warehouse fulfilment to this topic tree.
MATT.TOPIC.NEWS.INTERNAL	matt/news/internal/...	Matt's Deli publishes news of relevance to Matt's Deli stores and employees to this topic tree.
MATT.TOPIC.NEWS.EXTERNAL	matt/news/external/...	Matt's Deli publishes news of relevance to customers, suppliers and other external interested parties to this topic tree.

The table below describes the queue objects used in the scenario.

Table 12-2 Queue objects

Queue Name	Usage
MATT.RETAIL.ORDERS	Orders from stores and web customers are placed on this queue.
MATT.RETAIL.RESPONSES	Confirmations for received orders are sent as replies to this queue.
MATT.RETAIL.WH.ORDERS	Orders received from stores or web customers are placed on this queue for further processing before being sent on for fulfilment by a warehouse.
MATT.WH.ORDERS	Orders for fulfilment by a warehouse are placed on this queue. In the scenario this is a **QREMOTE** definition on the Headquarters system to a **QLOCAL** on a warehouse queue manager.

12.3 Components

The scenario is divided into a number of distinct components which are covered in the forthcoming chapters. Each component can be set up independently or they can be combined together into the complete scenario.

The remainder of this chapter provides a short overview of the purpose and flow of each component of the scenario.

12.3.1 Supplier pricing

This component uses retained publications and the different types of subscriptions supported by the Publish/Subscribe engine. This usage is illustrated by programs written in C and using the MQI.

Full details of the design and the operation of this component can be found in Chapter 14., "Scenario: Supplier Pricing using Pub/Sub" on page 267.

The relationship with the suppliers is managed centrally at headquarters. When a supplier is selected to be a provider of goods for Matt's Deli, they are given access to the supplier pricing system based on a WebSphere MQ V7.0 queue manager located at Matt's Deli headquarters.

When Matt's Deli needs to buy more of a particular product, a *request for quote* message is published to the appropriate point in the topic tree which has its root at *matt/requestquote/cat.* All suppliers who are subscribing to part of the tree at this point or above will receive a copy of the message.

Suppliers who wish to supply this product should publish a similar *price quote* message to the equivalent place in the topic tree which has its root at *matt/requestquote/cat*.

An illustrative flow for this component is shown below.

- Matt's Deli headquarters has a durable subscription to *matt/supplierquote/cat#* to ensure it receives all quotes from suppliers.
- Freds Fruit Farm is a supplier that supplies only fruit to Matt's Deli and subscribes to the topic string *matt/requestquote/cat/fresh/fruit/#.*
- Sams Super Supplier is a supplier that might be able to supply any product to Matt's Deli and subscribes to the topic string *matt/requestquote/cat/#.*
- Mikes Meat Market is a supplier that supplies only meat to Matt's Deli and subscribes to *matt/requestquote/cat/fresh/meat/#.*

- Matt's Deli needs a price for red apples so publishes a request quote message to *matt/requestquote/cat/fresh/fruit/apple/red.*
- The message is made available to Freds Fruit Farm and Sams Super Supplier but NOT Mikes Meat market.
- Sams Super Supplier chooses to quote and sends a price quote message for a price of ten pounds to *matt/supplierquote/cat/fresh/fruit/apple/red*.
- Matt's Deli receives the message. As it is cheaper than our current price Sams Super Supplier becomes our preferred supplier and we publish a modified retail price.
- Freds Fruit Farm chooses to quote for red apples too and sends a quote message priced at nine pounds.
- Matt's Deli receives the message, as it is cheaper than our current price Freds Fruit Farm becomes our preferred supplier and we publish a modified retail price.

12.3.2 Store ordering

This component uses Java Message Service(JMS) to send and receive the messages and illustrates the retained message feature and the use of the "+" symbol as a wildcard. This application uses both Publish/Subscribe and point to point features.

Full details of the design and the operation of this component can be found in Chapter 15., "Scenario: Store ordering with JMS" on page 283.

The product catalog required for the stores to place the orders is held as a topic tree with its root at *matt/retail/cat*. Each point in the tree has a retained publication message which contains pricing information. The store ordering application navigates up and down the tree by using the "+" (single topic level) wildcard. Having navigated to a product that is to be ordered, a point-to-point order message is sent to the MATT.RETAIL.ORDERS queue.

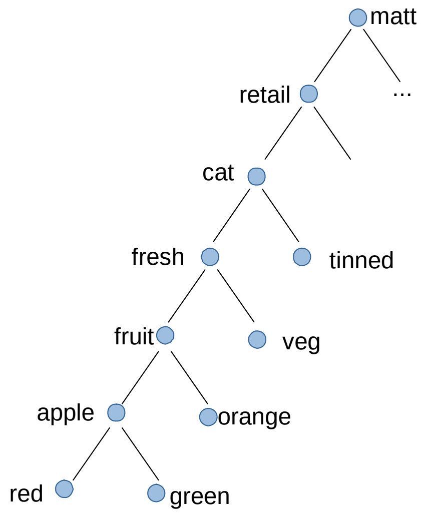


Figure 12-3 Sample fragment of retail catalog

An illustrative flow for this component is shown below, making use of the sample retail catalog in Figure 12-3.

- Store A of Matt's Deli needs to order more red apples so they start the store ordering application.
- The application subscribes to *matt/retail/cat/+* and receives just those retained publications for topic strings exactly one level below exactly one level below *matt/retail/cat/+*.
- In this case fresh and tinned are returned. The data in the messages identifies these as categories not products and so they are displayed as hierarchy choices.
- Selecting fresh causes the application to subscribe to topic string *matt/retail/cat/fresh/+*, this time returning fruit and veg.
- Selecting fruit causes the application to subscribe to topic string *matt/retail/cat/fresh/fruit/+*, this time returning apple and orange.
- Selecting apple causes the application to subscribe to topic string *matt/retail/cat/fresh/fruit/apple+*, this time returning red and green. Because these messages are identified as products the program displays prices and descriptions.
- Selecting red allows a quantity to be entered for an order, which is sent to the MATT.RETAIL.ORDERS queue for processing at headquarters.

12.3.3 News

Chapter 16., “Scenario: News using Client” on page 301

Author Comment: This section needs to be written when I understand News. Brian Cuttell

12.3.4 Web ordering

Author Comment: This section needs to be written in light of HTTP listener developments. Brian Cuttell

Chapter 17., “Scenario: Web ordering over HTTP” on page 319

12.3.5 Warehousing

This component uses the MQI to illustrate the new call back feature for asynchronous consumption of messages. In the scenario the warehouse is connected to a queue manager running on z/OS and the warehouse code is provided in both C and COBOL Languages. The warehouse application has been in place for some time and uses only point to point messaging. The scenario shows how an administrative subscription, created using MQ Explorer, can be used to subscribe on behalf of the warehouse and direct messages to an appropriate queue.

Full details of the design and the operation of this component can be found in Chapter 18., “Scenario: Warehousing using call back” on page 327.

An illustrative flow for this component is shown below.

- Store A has placed an order for red apples. This order is processed by headquarters and placed on the queue MATT.RETAIL.WH.ORDERS for fulfillment.
- Periodically (perhaps daily or hourly) a program at headquarters runs to publish all these orders to a topic tree with its root at *matt/warehouse/cat.* The red apple order is published here.
- For each warehouse (there is only one at the moment but Matt’s Deli is planning to grow) an administrative subscription automatically places messages on the relevant queue for the warehouse (when expanded to

multiple warehouses, these subscriptions become more complex). The red apple order is put on the warehouse queue by the administrative subscription. The MATT.WH.ORDERS queue is defined on the headquarters queue manager as a QREMOTE.

- The red apple message is now transmitted to the warehouse systems queue manager.
- On the warehouse system the warehousing fulfilment application is waiting on an asynchronous message consume. As soon as the red apple message arrives the message consumer (call back) code is invoked to process the red apple order message. In this simplified scenario the message is just displayed by the application rather than performing any business logic.

13

Scenario preparation

This chapter describes the common scenario setup which have to be perform before running any particular scenario components from the specific scenario chapters.

The following topics are covered:

- "Environment setup"
- "WebSphere MQ objects setup"

Any other setup and configuration tasks which are specific to a particular scenario component are described under the scenario component chapters.

13.1 Environment setup

In this section the environment used for scenario is described. The following topics are discussed:

- The logical topology of the scenario environment
- The physical topology of the scenario environment
- Machines configuration and software installation

13.1.1 The logical topology of the scenario environment

The following Figure shows the logical topology of scenario environment (WebSphere MQ topology):

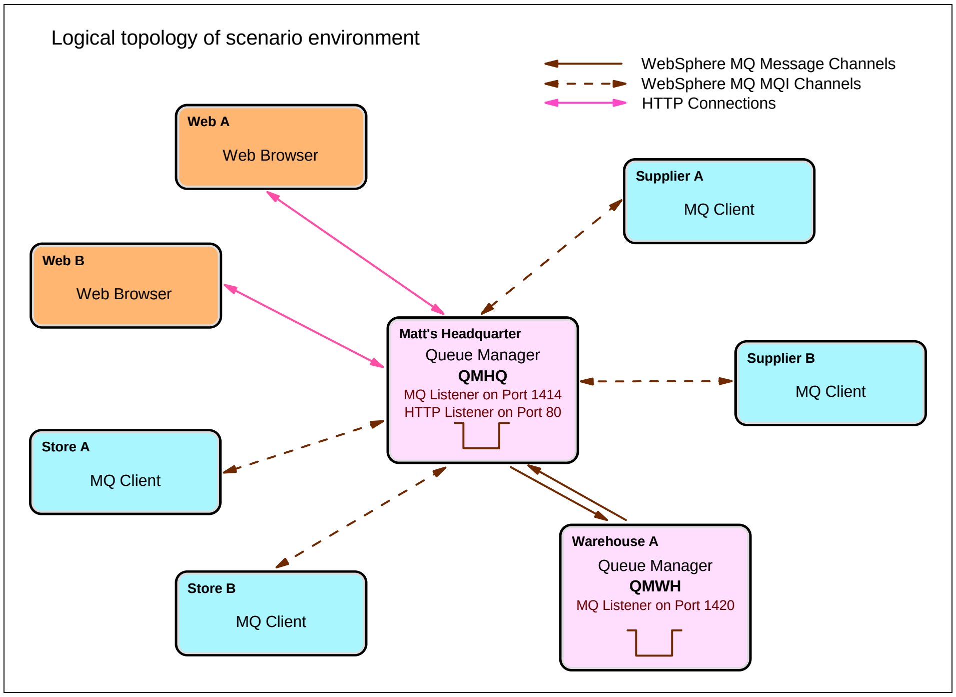


Figure 13-1 Logical topology of scenario environment

The environment consists of two queue managers, four MQ Clients and two client's PC with Web browser for Web access. All required queue managers and their connections are listed in this table:

Table 13-1 Required queue managers and their connections

Queue manager	Listener	Channel type	Channel usage
QMHQ	1414	Sender Receiver Server-connection	To QMWH From QMWH Application connections
QMWH	1420	Sender Receiver	To QMHQ From QMHQ

13.1.2 The physical topology of the scenario environment

The following Figure shows the physical topology of scenario environment:

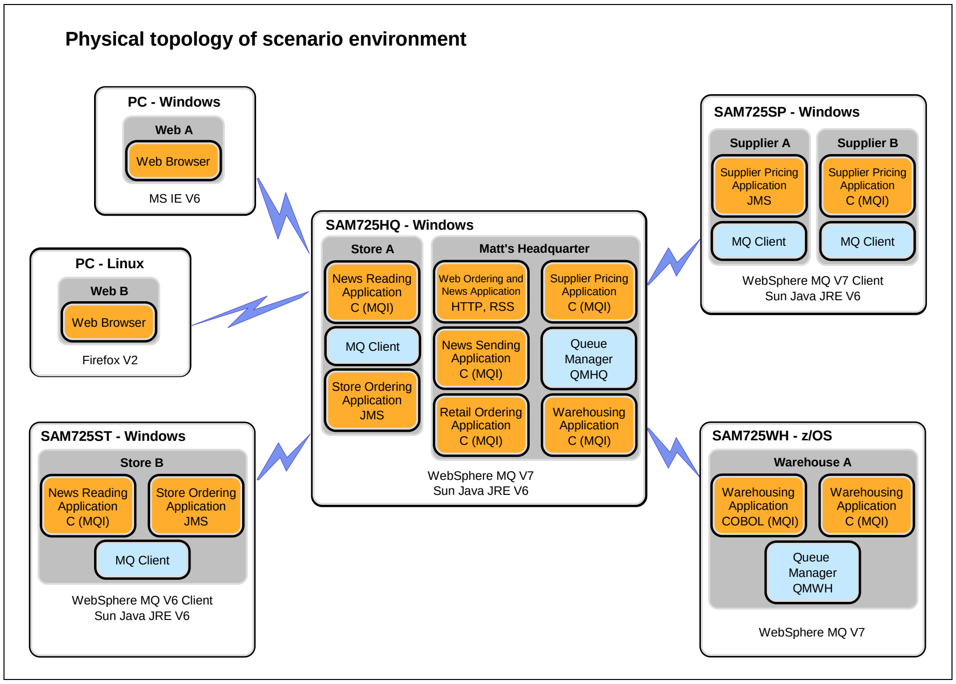

Physical topology of scenario environment
PC - Windows
Web A
Web Browser
MS IE V6
PC - Linux
Web B
Web Browser
Firefox V2
SAM725ST - Windows
Store B
News Reading Application C (MQI)
Store Ordering Application JMS
MQ Client
WebSphere MQ V6 Client
Sun Java JRE V6
SAM725HQ - Windows
Store A
News Reading Application C (MQI)
MQ Client
Store Ordering Application JMS
Matt's Headquarter
Web Ordering and News Application HTTP, RSS
Supplier Pricing Application C (MQI)
News Sending Application C (MQI)
Queue Manager QMHQ
Retail Ordering Application C (MQI)
Warehousing Application C (MQI)
WebSphere MQ V7
Sun Java JRE V6
SAM725SP - Windows
Supplier A
Supplier Pricing Application JMS
MQ Client
Supplier B
Supplier Pricing Application C (MQI)
MQ Client
WebSphere MQ V7 Client
Sun Java JRE V6
SAM725WH - z/OS
Warehouse A
Warehousing Application COBOL (MQI)
Warehousing Application C (MQI)
Queue Manager QMWH
WebSphere MQ V7

Figure 13-2 Physical topology of scenario environment

The physical environment consists of four machines that run Windows and two client's PC that run Windows and Linux operating systems. All software products used, and their versions, are listed in next tables:

Table 13-2 SAM725HQ - Windows machine for headquarters

Software	**Installed level**
Operating system	Microsoft Windows XP Service Pack 2
WebSphere MQ	V7.0
Java JRE™	Sun™ Java™ SE Runtime Environment 1.6.0_03

Table 13-3 SAM725SP - Windows machine for suppliers

Software	**Installed level**
Operating system	Microsoft Windows XP Service Pack 2
WebSphere MQ	V7.0 (Client only)
Java JRE	Sun Java SE Runtime Environment 1.6.0_03

Table 13-4 SAM725ST - Windows machine for store B

Software	**Installed level**
Operating system	Microsoft Windows XP Service Pack 2
WebSphere MQ	Client V6.0.2.2 (SupportPac MQC6)
Java JRE	Sun Java SE Runtime Environment 1.6.0_03

Table 13-5 SAM725WH - z/OS machine for warehouse

Software	**Installed level**
Operating system	z/OS V1.9
WebSphere MQ	V7.0

Table 13-6 PC - Windows machine for web client

Software	**Installed level**
Operating system	Microsoft Windows XP Service Pack 2
Web browser	Microsoft Internet Explorer® 6.0.2

Table 13-7 PC - Linux machine for web client

Software	Installed level
Operating system	openSUSE 10.3 (i586)
Web browser	Mozilla Firefox 2.0.0.6

Tip: All machines listed above are not required to run whole scenario. For example, it is possible to use only one machine and install all required components on it.

The recompiling of scenario program sources may be required to run on other platforms.

13.1.3 Machines configuration and software installation

This section describes basic machines configuration and installation of required software.

The installation of operating system is not covered in this chapter. Appropriate machines with supported operating systems are prerequisite for this scenario.

Note: For information about supported platforms for WebSphere MQ V7.0, refer to the *WebSphere MQ System Requirements*, at IBM web page:

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

Basic machines setup

The following tasks need to be done as prerequisites for the next steps:

- Check the user permissions: the administrator permissions are needed for machines configuration and software installations.
- Set up TCP/IP address and hostname: the TCP/IP access is required among scenario machines to MQ listeners TCP/IP ports, as described in Table 13-1 on page 253 and Figure 13-2 on page 253 above.

Creating groups and user IDs

The groups required for scenario need to be created on the machines as described in the table below:

Table 13-8 List of group IDs

Machine	Group ID	Use for
SAM725HQ	hq st sp matt	Headquarters applications Store applications and news Supplier access Public news
SAM725SP	sp matt	Supplier applications Public news
SAM725ST	st matt	Store applications and news Public news
SAM725WH	wh	Warehouse application

Then, the following users need to be created on the machines and put into appropriate groups as described in the table below:

Table 13-9 List of user IDs

Machine	User ID	Group ID	Use for
SAM725HQ	hquser stora storb suppa suppb	hq matt, st matt, st matt, sp matt, sp	Headquarters applications Stores A applications and access Store B access Supplier A access Supplier B access
SAM725SP	suppa suppb	matt, sp matt, sp	Supplier A applications Supplier B applications
SAM725ST	storb	matt, st	Store B applications
SAM725WH	whuser	wh	Warehouse application

None of the user is a member of `mqm` group to enable proper security policy.

WebSphere MQ V7.0 product installation

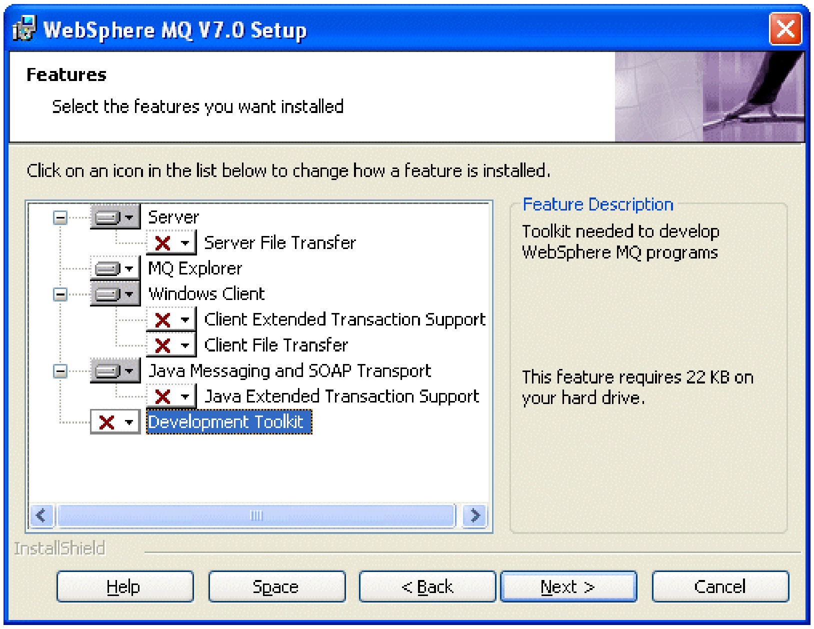
Perform the WebSphere MQ V7.0 product installation in accordance with the product installation guides, available online at WebSphere MQ V7.0 Information Center at IBM web page:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

The product is installed on three machines, as described in the following sections:

WebSphere MQ server installation on headquarters machine

Perform the WebSphere MQ server installation on headquarters machine, `SAM725HQ`. Select the **Custom** installation and then Server, MQ Explorer, Windows Client and Java Messaging and SOAP Transport components, as described on the figure below:



Select the features you want installed

Click on an icon in the list below to change how a feature is installed.

- Server
  - Server File Transfer
- MQ Explorer
- Windows Client
  - Client Extended Transaction Support
  - Client File Transfer
- Java Messaging and SOAP Transport
  - Java Extended Transaction Support
- Development Toolkit

Feature Description

Toolkit needed to develop WebSphere MQ programs

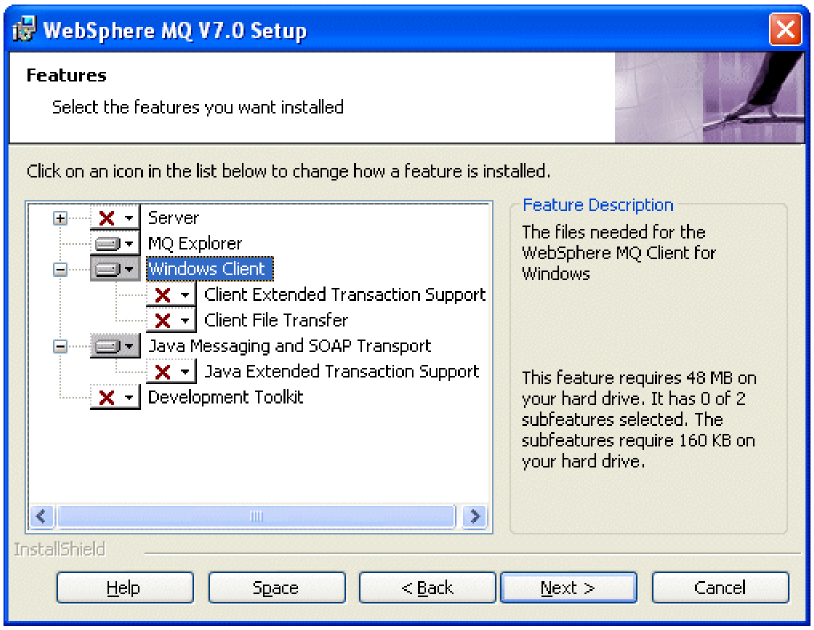
This feature requires 22 KB on your hard drive.

Figure 13-3 WebSphere MQ server installation

Use defaults for any other parameters or settings during the installation dialog.

WebSphere MQ client installation on suppliers machine

Perform the WebSphere MQ client installation on suppliers machine, `SAM725SP`. Select the **Custom** installation and then MQ Explorer, Windows Client and Java Messaging and SOAP Transport components, as described on the figure below:



Features

Select the features you want installed

Click on an icon in the list below to change how a feature is installed.

- Server
- MQ Explorer
- Windows Client
  - Client Extended Transaction Support
  - Client File Transfer
- Java Messaging and SOAP Transport
  - Java Extended Transaction Support
- Development Toolkit

Feature Description

The files needed for the WebSphere MQ Client for Windows

This feature requires 48 MB on your hard drive. It has 0 of 2 subfeatures selected. The subfeatures require 160 KB on your hard drive.

Figure 13-4 WebSphere MQ client installation

Use defaults for any other parameters or settings during the installation dialog.

WebSphere MQ server installation on warehouse machine

Perform the WebSphere MQ server installation on warehouse machine, `SAM725WH`. The server component installation is performed via standard SMP/E installer.

WebSphere MQ V6.0 client installation on store B machine

Download and install WebSphere MQ SupportPac MQC6 for WebSphere MQ V6.0 client installation on store B machine, `SAM725ST`. It is available at IBM web page:

`http://www.ibm.com/support/docview.wss?rs=171&uid=swg24009961&loc=en_US&cs=utf-8&lang=en`

Select the **Typical** installation and use defaults for any other parameters or settings during the installation dialog.

Java runtime environment installation

Download and install the latest Java runtime code (version 1.6.0_03 in the time of writing), available at Sun Java web site:

http://www.java.com

The Java JRE is used on all windows machines, headquarters `SAM725HQ`, suppliers `SAM725SP` and store B `SAM725ST`.

13.2 WebSphere MQ objects setup

This section contains list of MQ objects required for scenario:

- Queue managers
- Queue manager objects
  - Listeners
  - Channels
  - Queues
  - Topics and Subscriptions
- Object authorities (OAM)

The scripts to create all required objects are provided, refer to Appendix A, “Scenario preparation scripts” on page 337 for printed version of scripts and to Appendix B, “Additional material” on page 343 for scripts download.

Tip: All queue managers listed below are not required to run whole scenario. It is possible to use only one queue manager and create all required objects on it.

The scripts which are supplied for the scenario have to be modified to run on one queue manager.

13.2.1 Creating the queue managers

The following queue managers need to be created with appropriate operating system control command:

Table 13-10 List of queue managers

Machine	QM name	Dead letter queue
SAM725HQ	QMHQ	SYSTEM.DEAD.LETTER.QUEUE
SAM725WH	QMWH	SYSTEM.DEAD.LETTER.QUEUE

Note: The default dead letter queue setup for queue managers is covered in the MQSC scripts *QMHQobjects.txt* and *QMWHobjects.txt*, which are described in the next section.

13.2.2 Creating the queue managers objects

The queue manager objects need to be created by MQSC commands with the following scripts:

- *QMHQobjects.txt*: object definition MQSC commands for headquarters queue manager. To run this script issue:

  ```
  runmqsc QMHQ < QMHQobject.txt
  ```

  Tip: Verify commands in the script without performing the actions with parameter **-v**:

  ```
  runmqsc -v QMHQ < QMHQobject.txt
  ```

- *QMWHobjects.txt*: object definition MQSC commands for warehouse queue manager. Use the batch utility CSQUTIL program in case of z/OS.

These scripts use REPLACE parameter and can be run repeatedly. Refer to Appendix A, “Scenario preparation scripts” on page 337 for printed version of scripts and to Appendix B, “Additional material” on page 343 for scripts download.

Tip: WebSphere MQ Explorer can be used instead of MQSC commands. Create all objects from the tables below with appropriate MQ Explorer tasks.

The listeners

The listeners required for whole scenario are illustrated on the table below:

Table 13-11 List of listeners

QM name	Listener name	Port number	Controlled by
QMHQ	QMHQ.TCP	1414	Queue manager
QMHQ	QMHQ.HTTP	80	Queue manager
QMWH	QMWH.TCP	1420	Queue manager

The listeners are related to the following scenario chapters:

- The MQ listener QMHQ.TCP is related to the whole scenario, except the news scenario described in Chapter 16., “Scenario: News using Client” on page 301.
- The HTTP listener QMHQ.HTTP is related to the web ordering scenario described in Chapter 17., “Scenario: Web ordering over HTTP” on page 319 and to the news scenario described in Chapter 16., “Scenario: News using Client” on page 301.
- The MQ listener QMWH.TCP is related to the warehousing scenario described in Chapter 18., “Scenario: Warehousing using call back” on page 327.

Author Comment: The red text about HTTP listener have to be reviewed when the integrated HTTP listener is available.

Tip: To use only one queue manager for whole scenario create only the one MQ listener with port number 1414.

The channels

The channels required for whole scenario are illustrated on the table below:

Table 13-12 List of channels

QM name	Channel name	Channel type	Conn name	Xmit Q
QMHQ	QMHQ_TO_QMWH QMWH_TO_QMHQ QMHQ_SUPP_A QMHQ_SUPP_B QMHQ_STORES QMHQ_NEWS	Sender Receiver Server-conn Server-conn Server-conn Server-conn	sam725wh(1420)	QMWH
QMWH	QMWH_TO_QMHQ QMHQ_TO_QMWH	Sender Receiver	sam725hq(1414)	QMHQ

The channels are related to the following scenario chapters:

- The all sender and receiver channels are related to the warehousing scenario described in Chapter 18., “Scenario: Warehousing using call back” on page 327.
- The server-connection channels QMHQ_SUPP_A and QMHQ_SUPP_B are related to the supplier pricing scenario described in Chapter 14., “Scenario: Supplier Pricing using Pub/Sub” on page 267.

- The server-connection channel QMHQ_STORES is related to the store ordering scenario described in Chapter 15., “Scenario: Store ordering with JMS” on page 283 and to the news scenario described in Chapter 16., “Scenario: News using Client” on page 301.
- The server-connection channels QMHQ_NEWS is related to the news scenario described in Chapter 16., “Scenario: News using Client” on page 301.

Tip: To use only one queue manager for whole scenario the all sender and receiver channels are not needed.

The queues

The queues required for whole scenario are illustrated on the table below:

Table 13-13 List of queues

QM name	Queue name	Queue type	Parameters
QMHQ	QMWH	Local	USAGE: Transmission
	MATT.RETAIL.ORDERS	Local	
	MATT.RETAIL.RESPONSES	Local	
	MATT.RETAIL.WH.ORDERS	Local	PROPCTL: None
	MATT.SUPPLIER.QUOTES	Local	DEFPSIST: Yes DEFSOPT: Shared
	MATT.WH.ORDERS	Remote	RNAME: MATT.WH.ORDERS RQMNAME: QMWH
QMWH	QMHQ	Local	Usage: Transmission
	MATT.WH.ORDERS	Local	

The queues are related to the following scenario chapters:

- The transmission usage queues QMWH and QMHQ are related to the warehousing scenario described in Chapter 18., “Scenario: Warehousing using call back” on page 327.
- The local queues MATT.RETAIL.ORDERS and MATT.RETAIL.RESPONSES are related to the store ordering scenario described in Chapter 15., “Scenario: Store ordering with JMS” on page 283 and to the web ordering scenario described in Chapter 17., “Scenario: Web ordering over HTTP” on page 319.
- The local queue MATT.RETAIL.WH.ORDERS is related to the store ordering scenario described in Chapter 15., “Scenario: Store ordering with JMS” on page 283, to the web ordering scenario described in Chapter 17., “Scenario: Web ordering over HTTP” on page 319 and to the warehousing scenario described in Chapter 18., “Scenario: Warehousing using call back” on page 327.

- The local queue MATT.RETAIL.WH.ORDERS is related to the supplier pricing scenario described in Chapter 14., “Scenario: Supplier Pricing using Pub/Sub” on page 267.
- The remote and local queues MATT.WH.ORDERS are related to the warehousing scenario described in Chapter 18., “Scenario: Warehousing using call back” on page 327.

Tip: To use only one queue manager for whole scenario the all transmission usage queues and MATT.WH.ORDERS remote queue are not needed.

The topics and subscriptions

The topics and subscriptions required for whole scenario are illustrated on the two tables below:

Table 13-14 List of topics

QM name	Topic name	Topic string	Parameters
QMHQ	MATT.TOPIC.REQUESTQUOTE MATT.TOPIC.SUPPLIERQUOTE MATT.TOPIC.WAREHOUSE MATT.TOPIC.RETAIL MATT.TOPIC.NEWS.INTERNAL MATT.TOPIC.NEWS.PUBLIC	matt/requestquote matt/supplierquote matt/warehouse matt/retail matt/news/internal matt/news/public	DEFPSIST: Yes
QMWH	N/A	N/A	

Table 13-15 List of subscriptions

QM name	Subscription name	Topic object	Parameters
QMHQ	MATT.SUB.WH.ORDERS	MATT.TOPIC. WAREHOUSE	Topic string: # DEST: MATT.WH.ORDERS DESTCLAS: PROVIDED
QMWH	N/A	N/A	

The topics and subscriptions are related to the following scenario chapters:

- The topics MATT.TOPIC.REQUESTQUOTE and MATT.TOPIC.SUPPLIERQUOTE are related to the supplier pricing scenario described in Chapter 14., “Scenario: Supplier Pricing using Pub/Sub” on page 267.
- The topic MATT.TOPIC.WAREHOUSE is related to the warehousing scenario described in Chapter 18., “Scenario: Warehousing using call back” on page 327.

- The topic MATT.TOPIC.RETAIL is related to the supplier pricing scenario described in Chapter 14., “Scenario: Supplier Pricing using Pub/Sub” on page 267, to store ordering scenario described in Chapter 15., “Scenario: Store ordering with JMS” on page 283 and to the web ordering scenario described in Chapter 17., “Scenario: Web ordering over HTTP” on page 319.
- The topics MATT.TOPIC.NEWS.INTERNAL and MATT.TOPIC.NEWS.PUBLIC are related to the news scenario described in Chapter 16., “Scenario: News using Client” on page 301.
- The subscription MATT.SUB.WH.ORDERS is related to the supplier pricing scenario described in Chapter 14., “Scenario: Supplier Pricing using Pub/Sub” on page 267.

Tip: There are no administered topic objects at QMWH queue manager so there is no impact to use only one queue manager for whole scenario.

13.2.3 Setting object authorities (OAM)

The appropriate authorities for headquarters queue manager objects need to be set by operational command **`setmqaut`**. The script **`QMHQsetaut.bat`** can be used.

The script removes all authorities at first and can be run repeatedly. Refer to Appendix A, “Scenario preparation scripts” on page 337 for printed version of script and to Appendix B, “Additional material” on page 343 for script download.

The appropriate authorities for warehouse queue manager objects need to be set at RACF subsystem on z/OS.

Tip: WebSphere MQ Explorer can be used instead of MQSC commands. Create all objects from the tables below with appropriate MQ Explorer tasks.

Table 13-16 List of authorities

QM name	GID	Object name	Object type	Authority
QMHQ	hq	QMHQ	Queue manager	Connect
		MATT.RETAIL.ORDERS	Queue	Get
		MATT.RETAIL.RESPONSES	Queue	Put
		MATT.RETAIL.WH.ORDERS	Queue	Get, Put
		MATT.WH.ORDERS	Queue	Put
		MATT.TOPIC.REQUESTQUOTE	Topic	Pub
		MATT.TOPIC.SUPPLIERQUOTE	Topic	Sub
		MATT.TOPIC.WAREHOUSE	Topic	Pub, Sub
		MATT.TOPIC.RETAIL	Topic	Pub, Sub
		MATT.TOPIC.NEWS.*	Topic	Pub
	st	QMHQ	Queue manager	Connect
		MATT.RETAIL.ORDERS	Queue	Put
		MATT.RETAIL.RESPONSES	Queue	Get
		MATT.TOPIC.RETAIL	Topic	Sub
		MATT.TOPIC.NEWS.INTERNAL	Topic	Sub
	sp	QMHQ	Queue manager	Connect
		MATT.TOPIC.REQUESTQUOTE	Topic	Sub
		MATT.TOPIC.SUPPLIERQUOTE	Topic	Pub
	matt	MATT.TOPIC.NEWS.PUBLIC	Topic	Sub
QMWH	wh	QMWH	Queue manager	Connect
		MATT.WH.ORDERS	Queue	Get

Tip: To use only one queue manager for whole scenario the authority settings for warehouse queue manager QMWH are not needed.

Author Comment: Due to the alpha version limitation we also have to set up the following authorities, which will only be available for GA code...

Table 13-17 Should be removed from GA version of book

QM name	GID	Object name	Object type	Authority
QMHQ	hq	SYSTEM.DURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq
		SYSTEM.NDURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq
	st	SYSTEM.DURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq
		SYSTEM.NDURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq
	sp	SYSTEM.DURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq
		SYSTEM.NDURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq
	matt	SYSTEM.DURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq
		SYSTEM.NDURABLE.MODEL.QUEUE	Queue	Browse, Get, Put, Inq

14 Scenario: Supplier Pricing using Pub/Sub

This component of the scenario illustrates the use of retained publications and the different types of subscriptions supported by the Publish/Subscribe engine.

This scenario demonstrates durable, non-durable, managed and non-managed subscriptions.

Retained publications are used to store the current retail prices of all the products offered by Matt's Deli. The topic hierarchy is used to classify the products for the price catalog.

The following topics are discussed:

- "Design overview"
- "Deploying the Supplier Pricing component"
- "Running the Supplier Pricing component"
- "Verifying the Supplier Pricing component"
- "Summary"

14.1 Design overview

Matt's Deli headquarters keeps a retail price catalog that is used by all the Deli stores to order new stock. The relationship with the suppliers is managed centrally at the headquarters. When a supplier is selected to be a provider of goods for Matt's Deli, they are given access to the supplier pricing system. This system broadcasts to all suppliers requests for price quotes when Matt's Deli needs to buy more products. Suppliers may reply with a price quote for the products that they can supply. Matt's Deli receives and processes all the price quotes from all suppliers. It selects the cheapest quote and presumably places a purchase order to the selected supplier. The goods should be delivered to the warehouse but the order processing is out of the scope of this scenario.

The Supplier Pricing system is based on a WebSphere MQ V7.0 queue manager located at Matt's Deli headquarters. The Supplier Pricing component uses the Publish/Subscribe engine to publish "request for quote" messages that are broadcasted to all suppliers that have MQ Client applications subscribed to the topic "matt/requestquote/cat/#".

Suppliers receive the catalog id and the description of the product required. Suppliers have the option to ignore the request or they can reply publishing a "price quote" message to the topic "matt/supplierquote/<catalog id>". The catalog id is a topic string that is concatenated to "matt/supplierquote".

Matt's Deli has an application program that subscribes to topic "matt/supplierquote/cat/#" to receive quotes from all suppliers. This application receive the supplier quote messages, it checks if the price is cheaper than a previous quote received from other supplier, it calculates a retail price (Matt's Deli business rule says that retail price is twice the cheapest supplier price) and it then publishes a retained publication to topic "matt/retail/<catlog id> (this is the retail price catalog). If the previous quote is from the same supplier for the same product the new quote is accepted as new price and published in the retail price catalog.

If the supplier quote is more expensive than the existing quote from a different supplier then the quote is discarded (rejected).

The retail price catalog is used by the stores and by the internet users to display products and prices before making purchase orders.

The following figure describes the programs and the interaction between Headquarters and the Suppliers:

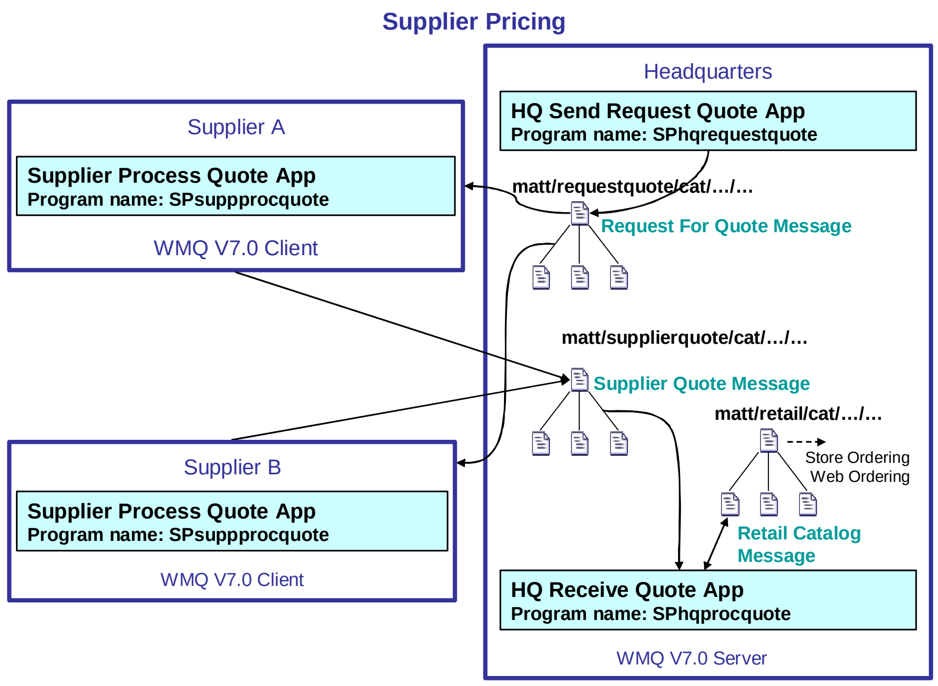

Supplier Pricing
Headquarters
HQ Send Request Quote App
Program name: SPhqrequestquote
Supplier A
Supplier Process Quote App
Program name: SPsuppprocquote
WMQ V7.0 Client
matt/requestquote/cat/…/…
Request For Quote Message
matt/supplierquote/cat/…/…
Supplier Quote Message
matt/retail/cat/…/…
Store Ordering
Web Ordering
Retail Catalog Message
Supplier B
Supplier Process Quote App
Program name: SPsuppprocquote
WMQ V7.0 Client
HQ Receive Quote App
Program name: SPhqprocquote
WMQ V7.0 Server

Figure 14-1 Supplier Pricing component

14.2 Deploying the Supplier Pricing component

The supplier pricing component is made of three application programs. Two of the programs run locally on the headquarter's (HQ) computer system and the third one is a remote client program that runs on the supplier's computer system. Each supplier may run a copy of the client program provided by Matt's Deli or a copy of a client application that has been developed by the supplier.

The following are programs that implement the supplier pricing component:

- **HQ send request for quote (SPhqrequestquote) -** This is program is used by the headquarters staff to send request for quote messages to all the suppliers.
- **HQ process supplier quotes (SPhqprocquote) -** This program runs on the headquarters to receive supplier quotes, to decide if the price quoted is acceptable and to publish a new retail price.

- **Supplier process quotes (SPsuppprocquote) -** This program is used by the suppliers to receive and respond to request for quote messages from Matt's Deli headquarters.

A WebSphere MQ V7.0 queue manager runs on the headquarters to provide the services of Pub/Sub engine for the supplier pricing component. The headquarters application programs on the same server as the queue manager and they connect to the queue manager using server bindings. Supplier application programs are MQ Client applications and they connect to the queue manager using client bindings.

14.2.1 HQ send request for quote program

This is a C language program runs on the same server as the queue manager and it interacts with Matt's Deli staff to send price quote requests to the suppliers. The main functions of the program are:

- It connects to the queue manager. The queue manager name can be specified as first parameter (argv[1]) to the program. If the queue manager name is not specified the program tries to connect to the default queue manager.
- It asks the HQ user to enter the catalog id and the product description from the keyboard (stdin). The catalog id is the topic string that is concatenated to the topic string specified in the topic object MATT.TOPIC.REQUESTQUOTE to form the topic for the publication.
- It builds a request for quote message. This message has two comma separated fields in the message body: <catalog id>,<product description>.
- It uses MQPUT1 to publish the request for quote to the topic.
- It repeats asking for a new request for quote until the user requests to terminate the program.

This program demonstrates how to make a publication to a topic concatenating the topic string specified in the topic object with the topic string received from the keyboard. It uses the MQPUT1 function call to publish to a topic object descriptor.

14.2.2 HQ process supplier quotes program

This C language program runs on the same server as the queue manager. The following are main functions of this program:

- It receives two optional parameters: the first parameter (argv[1]) is the subscription queue name and the second parameter (argv[2]) is the queue manager name. If the queue name is not specified it assumes the name MATT.SUPPLIER.QUOTES. If the queue manager name is not specified then the program connects to the default queue manager.

- It connects to the queue manager.
- It opens the subscription queue.
- It creates or resumes a non-managed durable subscription with the name MATT.SUB.SUPPLIERQUOTE. The topic is made of the concatenation of the topic string defined in the topic object MATT.TOPIC.SUPPLIERQUOTE and the topic string cat/#. The subscription is created the first time this program is executed and it remains active until it is removed by the administrator using MQ Explorer or a command. If this program is not active the publications are stored in the subscription queue.
- It loops calling MQGET to receive supplier quote messages from the topic. MQGET waits for two minutes and if no messages are received it asks the user if it should continue or terminate the program.
- For every message that is received a function ProcessAQuote is called to do the following process:
  - It creates a managed non-durable subscription to the retail price catalog with the topic string made of the catalog id of the product received in the supplier quote. The subscription allows to receive retained publications.
  - It receives a retained publication using MQGET. Matt's Deli does not allow non retained publications on the retail price catalog, therefore no other type of publications are expected.
  - If a retained publication is not received then this is a new product, therefore a function PublishRetailCatalogItem is called to create and publish a retail price catalog message. For new products retained publications are published for each level of the topic string. The retail price catalog has a four level topic string: "cat/<product category>/<product type>/<product name>", therefore one message is published for cat, another for product category and another for product type. These additional retained publications enable Matt's Deli stores to navigate and discover the contents of the retail price catalog.
  - If a retained publication is received then this is the existing catalog entry with the current retail price, supplier price and supplier name for this product. The current supplier price is compared with the price quoted in the supplier's message.
  - If the quote is from the same supplier as the current supplier this is considered a price change and it is published to the retail price catalog calling the function PublishRetailCatalogItem.

  - If the current supplier price is higher than the quoted price then this is a cheaper quote and it is published to the retail price catalog calling the function PublishRetailCatalogItem.
  - If the current price is lower or equal to the price quoted then the supplier quote is rejected and the retail price catalog is not updated.
- For every retail price catalog update a function PublishRetailCatalogItem is called:
  - It opens a topic object with a topic string that is the catalog id of the product.
  - It prepares a retail price publication. This message is a comma separated text with five elements: catalog id, product description, retail price, supplier price and supplier name.
  - It publishes the retail price message as retained publication calling MQPUT1.

This program demonstrates how to create and resume non-managed durable subscriptions. Supplier quotes are received even when this program is not active. It shows how to subscribe using managed non-durable subscriptions to get retained publications and also how to publish retained publications when it is accessing and updating the retail price catalog.

14.2.3 Supplier process quotes program

This is a C language program that runs as MQ Client in the supplier's computer system. This program interacts with the Supplier's staff to receive and respond Matt's Deli request for quotes. The following are the main functions of this program:

- It receives a queue manager name as the only parameter (argv[1]) passed to the program. If the queue manager name is not provided then a connection to the default queue manager is made.
- It asks the Supplier user for the supplier identification.
- It connects to the queue manager. The environment variable MQSERVER=<svrconn channel name>/tcp/<hostname>(<port>) must be defined before running this program.
- It creates a subscription using a topic object called MATT.TOPIC.REQUESTQUOTE and a object string cat/#. The resolved to topic string is the concatenation of the topic string defined in the topic object and the object string cat/#.

- It gets request for quote publications from the created subscription. If the MQGET call waits for more than 2 minutes without receiving a message then it asks the user if it should continue or to terminate the program.
- It calls function ProcessARequestQuote for each request for quote message received. This function does the following process:
  - It parses the received message to get the product catalog Id and the production description.
  - It displays to the user the request for quote received.
  - It asks the user for a price to quote or zeros if the request is to be ignored.
  - If zeros is entered the function returns without doing any further process to the request for quote.
  - If a price is entered then a supplier quote message is prepared.
  - It opens a topic object called MATT.TOPIC.SUPPLIERQUOTE and an object string that is the catalog id received from the request for quote.
  - It publishes a supplier quote to the topic using MQPUT1.
- It loops to get next request for quote message from Matt's Deli.

14.2.4 Required MQ objects

The Supplier Pricing component requires the following objects to be defined by the queue manager administrator:

- **MATT.TOPIC.REQUESTQUOTE -** This topic defines a topic string of matt/requestquote. The purpose of this object is to define access control such as Matt's Deli headquarters can publish and subscribe to the topic but suppliers can only subscribe.
- **MATT.TOPIC.SUPPLIERQUOTE -** This topic defines a topic string of matt/supplierquote and it also defines that publications to this topic should be persistent by default. The access control should allowed suppliers to publish but no to subscribe to this topic. Headquarters should be able to subscribe and publish.
- **MATT.SUPPLIER.QUOTES -** This is the subscriber queue for the non-managed durable subscription created by the HQ process supplier quotes program. This queue stores all supplier quotes while the program is busy or not active.

The following are the commands to create these objects:

Example 14-1 Supplier Pricing topic and queue objects

```
DEFINE TOPIC(MATT.TOPIC.SUPPLIERQUOTE) +
   TOPICSTR('matt/supplierquote') +
   REPLACE +
   DEFPSIST(YES) +
   PUB(ENABLED) +
   SUB(ENABLED)

DEFINE TOPIC(MATT.TOPIC.SUPPLIERQUOTE) +
   TOPICSTR('matt/requestquote') +
   REPLACE +
   PUB(ENABLED) +
   SUB(ENABLED)

DEFINE QLOCAL(MATT.SUPPLIER.QUOTES) +
   LIKE(SYSTEM.DEFAULT.LOCAL.QUEUE) +
   REPLACE +
   DEFPSIST(YES) +
   DEFSOPT(SHARED)
```

14.2.5 Installation of Supplier Pricing component

The Supplier Pricing component programs are packaged in a file called CH14_SupplierPricing.zip available for download in Appendix B, "Additional material" on page 343.

The following are the instructions on how to install the executable code:

- Extract the contents of the compressed file into a temporary directory, for example C:\temp.
- The following directories are created:
  - C:\temp\SupplierPricing\SPhqprocquote\
  - C:\temp\SupplierPricing\SPhqrequestquote\
  - C:\temp\SupplierPricing\SPsuppprocquote\

  There are two additional directories with other programs that can be used to test this component:
  - C:\temp\SupplierPricing\loader\
  - C:\temp\SupplierPricing\SPsuppsendquote\
- Each directory contains the following files:
- **<program name>.c -** This is the program the source code.

- **<program name>.exe -** This is the executable code linked with server bindings library.
- **<program name>c.exe -** This is the executable code linked with the client bindings library.
- **make.bat -** This the batch command file that executes nmake to compile and link the program. Execute this command if it necessary to recreate the executable code.
- **makefile.txt -** This is the nmake parameter file.

These programs are compiled with a C language compiler such as Microsoft Visual C++® Express Edition for Windows.

14.3 Running the Supplier Pricing component

To run the Supplier Pricing component programs, execute each program on a its own command prompt window.

14.3.1 HQ send request for quote program

The following are the instructions to run the SPhqrequestquote program:

- Open a Windows command prompt.
- Check that queue manager is running:

```
C:\>dspmq
QMNAME(WMQ7) STATUS(Running)
```

- Change directory to the location of the executable code.

```
C:\> cd C:\temp\SupplierPricing\SPhqrequestquote
```

- Execute the program:

```
C:\temp\SupplierPricing\SPhqrequestquote>SPhqrequestquote.exe
**************************************************************
* Welcome to MATTS DELI - Prepare and send request for quote **
**************************************************************

Enter the catalogID to request a quote for:
..  cat/<fresh|tinned|dry|glass>/<product type>/<product>
..  Example: 'cat/fresh/fruit/apple'
..  or press ENTER to end

--> CatalogId :
```

- Enter the catalog id for the product to request the quote:

```
--> CatalogId : cat/fresh/fruit/oranges

Enter product description:
..  (note no commas)

--> Description :
```

- Enter the product description:

```
Enter product description:
..  (note no commas)

--> Description : Spanish oranges per Kg

<*> Published message <cat/fresh/fruit/oranges,Spanish oranges per
Kg>

Enter the catalogID to request a quote for:
..  cat/<fresh|tinned|dry|glass>/<product type>/<product>
..  Example: 'cat/fresh/fruit/apple'
..  or press ENTER to end

--> CatalogId :
```

- Press Enter to end the program:

```
Enter the catalogID to request a quote for:
..  cat/<fresh|tinned|dry|glass>/<product type>/<product>
..  Example: 'cat/fresh/fruit/apple'
..  or press ENTER to end

--> CatalogId :
**************************************************************
* End of MATTS DELI prepare and send request for quote *******
***************** End Program ********************************
```

14.3.2 HQ process supplier quotes program

The following are the instructions to run the SPhqprocquote program:

- Open a Windows command prompt.
- Check that queue manager is running:

```
C:\>dspmq
QMNAME(WMQ7) STATUS(Running)
```

- Change directory to the location of the executable code.

```
C:\> cd C:\temp\SupplierPricing\SPhqprocquote
```

- Execute the program:

```
C:\temp\SupplierPricing\SPhqprocquote>SPhqprocquote.exe
Supplier Pricing Process Supplier Quote Application START
<*> Connected to Queue Manager:
<*> Open Subscription queue: <MATT.SUPPLIER.QUOTES>
<*> MQSUB ResolvedTopicString <matt/supplierquote/cat/#>
<*> Calling MQGET : 180 seconds wait time
```

- Wait for supplier quotes to arrive:

```
<*> Input supplier quote <cat/fresh/fruit/oranges,Spanish oranges
per Kg,0.65,SUPPA>

Processing an inbound quote
.. Catalog : cat/fresh/fruit/oranges
.. Desc    : Spanish oranges per Kg
.. Price   : 0.65
.. Supplier: SUPPA

New product publishing retained retail price for
cat/fresh/fruit/oranges of  1.30
<*> Target Topic is = matt/retail/cat/fresh/fruit/oranges
<*> Publish message <cat/fresh/fruit/oranges,Spanish oranges per Kg,
1.30,0.65,SUPPA>

<*> Target Topic is = matt/retail/cat
<*> Publish message <cat,cat,0.00,0.00,CATEGORY>

<*> Target Topic is = matt/retail/cat/fresh
<*> Publish message <cat/fresh,fresh,0.00,0.00,CATEGORY>

<*> Target Topic is = matt/retail/cat/fresh/fruit
<*> Publish message <cat/fresh/fruit,fruit,0.00,0.00,CATEGORY>

<*> Calling MQGET : 180 seconds wait time
```

- Enter quit to terminate:

```
<*> Calling MQGET : 180 seconds wait time
<*> No more messages
Press ENTER to continue or type QUIT to terminate.
```

quit

```
<*> Program terminating
<*> Closing subscription handle (KEEP)
<*> Closing managed destination handle
Supplier Pricing Process Supplier Quote Application END
```

14.3.3 Supplier process quotes program

The following are the instructions to run the SPhqrequestquote program:

- Open a Windows command prompt.
- Change directory to the location of the executable code.

```
C:\> cd C:\temp\SupplierPricing\SPsuppprocquote
```

- Set the MQSERVER environment variable to establish the connection information to the queue manager:

```
C:\temp\SupplierPricing\SPsuppprocquote>set MQSERVER=
QMHQ_SUPP_A/tcp/9.20.16.251(1414)
```

- Execute the program:

```
C:\temp\SupplierPricing\SPsuppprocquote>SPsuppprocquotec.exe WMQ7
**************************************************************
* Welcome to MATTS DELI receive request for quote          ***
* and send a supplier price quote.                         ***
**************************************************************

Enter your Supplier Identification
.. or press ENTER to end

--> SupplierId :
```

- Enter the supplier identification:

```
--> SupplierId : SUPPA
<*> Connecting to queue manager: WMQ7
<*> MQSUB ResolvedTopicString <matt/requestquote/cat/#>
<*> Calling MQGET : 180 seconds wait time
```

- Wait for requests to arrive from Matt's Deli:

```
<*> Received publication <cat/fresh/fruit/oranges,Spanish oranges
per Kg>
Received a request for quote from Matt's Deli
.. Catalog : cat/fresh/fruit/oranges
.. Desc    : Spanish oranges per Kg

Enter a price to quote to Matt's Deli
.. or enter 0.00 to skip this request.
.. price format 999.99
--> Price :
```

- Enter a price to send supplier quote or zeros to skip the request:

```
--> Price : 0.65

<*> Target Topic is = /matt/supplierquote/cat/fresh/fruit/oranges
<*> Published message <cat/fresh/fruit/oranges,Spanish oranges per
Kg,0.65,SUPPA>
<*> Calling MQGET : 180 seconds wait time
```

- Enter quit to terminate the program:

```
<*> Calling MQGET : 180 seconds wait time

<*> No more messages
Press ENTER to continue or type QUIT to terminate.
```

quit

```
Program terminating
<*> Closing subscription handle (KEEP)
MQCLOSE ended with reason code 2045
<*> Closing managed destination handle
***************************************************************
* End of MATTS DELI receive and process requests for quote ***
***************** End Program ********************************
```

14.4 Verifying the Supplier Pricing component

To verify the Supplier Pricing component, follow these steps:

- Start programs SPhqrequestquote and SPhqprocquote in separate command windows in the headquarters server where the queue manager is running.
- Start the program SPsuppprocquote on a command window in the supplier system as MQ Client application.
- Enter a request for quote on the SPhqrequestquote program.
- Receive the request on the SPsuppprocquote program. Respond generating a supplier quote entering a price that is not zeros.
- Observe the supplier quote being processed by the SPhqprocquote program.
- Use sample program amqssub to subscribe to the topic matt/retail/cat/# to receive the retained publications from the retail price catalog.

```
C:\>amqssub matt/retail/cat/#
```

```
Sample AMQSSUBA start
Calling MQGET : 30 seconds wait time
message <cat/freh/fruit/apple,cox apples per kg, 2.16,1.08,SUPPA>
Calling MQGET : 30 seconds wait time
message <cat/freh/fruit/oranges,spanish oranges per kg,
1.98,0.99,SUPPA>
Calling MQGET : 30 seconds wait time
message <cat/fresh/fruit,fruit,0.00,0.00,CATEGORY>
Calling MQGET : 30 seconds wait time
message <cat/frseh,fresh,0.00,0.00,CATEGORY>
Calling MQGET : 30 seconds wait time
message <cat/glass/wine/red,spanish rioja,10.70,5.35,SUPPA>
Calling MQGET : 30 seconds wait time
message <cat/glass/wine/white,german white,14.50,7.25,SUPPA>
Calling MQGET : 30 seconds wait time
message <cat/glass/wine,wine,0.00,0.00,CATEGORY>
Calling MQGET : 30 seconds wait time
message <cat/glass,glass,0.00,0.00,CATEGORY>
Calling MQGET : 30 seconds wait time
message <cat/fresh/fruit/oranges,Spanish oranges per Kg,
1.30,0.65,SUPPA>
Calling MQGET : 30 seconds wait time
message <cat,cat,0.00,0.00,CATEGORY>
Calling MQGET : 30 seconds wait time
no more messages
Sample AMQSSUBA end
```

14.5 Summary

The Matt’s Deli Supplier Pricing component uses Publish/Subscribe to broadcast request for price quotes to all the suppliers. The suppliers subscribe to Matt’s Deli requests and decide to reply with a price quote.

Matt’s Deli uses retained publications that is a feature of WebSphere MQ V7.0 Publish/Subscribe to implement the retail price catalog. Retained publications allow to maintain a single price for each product in the catalog. The classification of the products in the catalog is implemented by the topic hierarchy. Matt’s Deli stores can retrieve the current price of any product.

The the catalog is updated by publishing a new retained publication to the topic that represent a specific product.

The Supplier Pricing component demonstrates how to create durable and non durable subscriptions and how managed and non managed subscriptions work.

In summary, these programs show how to do Publish/Subscribe using the new Message Queue Interface extensions.

15

Scenario: Store ordering with JMS

This chapter discusses the store ordering scenario. This chapter briefs about how to setup the headquarters application to accept the requests and the also explain how to setup the Store ordering application to submit the orders to the headquarters.

This chapter includes the following sections:

15.1 Design overview

The product catalogues required for the stores to place the orders are generated and stored at the headquarters of Matt’s Deli. The product catalogue contains details of the products headquarters can supply to the stores. Product descriptions are stored as a retained publication in the WebSphere MQ.

The Store ordering application is independent of all other components of the scenario. However there are a couple of pre-requisites that are needed for the Store ordering application to function and are described below.

The product catalogues are stored as Retained publications in the WebSphere MQ at headquarters. The pre-requisite for the store order application to work is to ensure that the product catalogue is loaded. Refer section “15.4, “Deploying the Store ordering component” on page 288 for more information.

When the application starts, the application subscribes to the “matt/retail/cat/+” topic to fetch all the main product categories. The user then has the option to browse through all the product types and the product items for a selected category. The user can then submit the order for the product to headquarters. Once the user submits the order, headquarters sends a response to the application. The store order application displays the response received to the user. If the headquarters does not send a response, then the store ordering application waits for a maximum of 3 seconds. And the control is returned back to the user to place a new order. The user has an option to check for the responses asynchronously that were not processed by the Store ordering application.

Store ordering application uses Java Message Service for sending and receiving the messages. This application uses both Publish/Subscribe and point to point feature. Figure 15.1 shows the work flow of the Store ordering application.

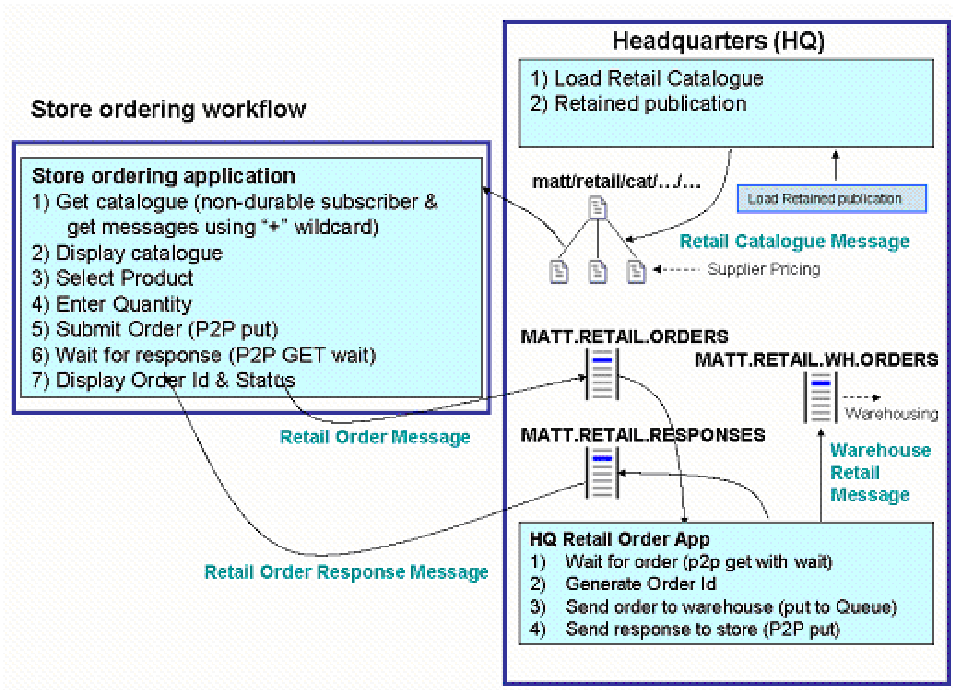

Store ordering workflow
Store ordering application
1) Get catalogue (non-durable subscriber & get messages using "+" wildcard)
2) Display catalogue
3) Select Product
4) Enter Quantity
5) Submit Order (P2P put)
6) Wait for response (P2P GET wait)
7) Display Order Id & Status
Retail Order Message
Retail Order Response Message
Headquarters (HQ)
1) Load Retail Catalogue
2) Retained publication
matt/retail/cat/…/…
Load Retained publication
Retail Catalogue Message
Supplier Pricing
MATT.RETAIL.ORDERS
MATT.RETAIL.WH.ORDERS
Warehousing
MATT.RETAIL.RESPONSES
Warehouse Retail Message
HQ Retail Order App
1) Wait for order (p2p get with wait)
2) Generate Order Id
3) Send order to warehouse (put to Queue)
4) Send response to store (P2P put)

Figure 15-1 StoreOrdering component

15.2 Deploying the headquarters process application

Both the JMS and the Web store ordering applications sends their request to the headquarters to place the order for the products. When the headquarters receives the order, the JMS application running on the headquarters sends the request to the Warehouse and also sends a response to the store.

The headquarters application is a JMS application and must always be running connecting to the headquarters queue manger where the stores sends the requests.

Before the headquarters application could be started, the following scripts must be run on the machine were the headquarters application would run.

1. Ensure WebSphere MQ client is installed on the machine where the Store Ordering application runs.
2. Ensure Java Development Kit 1.4.2 or higher is installed and configured on the machine.

3. The classpath variable must be set to include all the WMQ v7.0 jar files found under the <WMQ Home>/java/lib directory:

Example 15-1 Set classpath variable

```
set classpath=.;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\CL3Export.zip;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\CL3Nonexport.zip;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\com.ibm.mqbind.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\connector.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\fscontext.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\jndi.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\jta.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\ldap.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\mqjbdf02.dll;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\mqjbnd05.dll;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\MQXAi02.dll;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\providerutil.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\rmm.jar;
```

4. Extract the HQProcess.zip file to the c:\HQProcess directory.

> **Note:** HQProcess.zip - contains all the files required for the headquarters application to process the incoming requests.
>
> hqprocess.jar – The main jar file that contains the class files.
>
> HQSetup.bat – Initial setup file for the headquarters application.

5. Modify the classpath to include the hqprocess.jar reference at the start of the classpath as shown below

```
set classpath=c:\HQProcess\hqprocess.jar;%classpath%;
```

6. Edit the HQSetup.bat file and modify the following values

```
CUR_DIR - Set this value to the current working directory
```

Example 15-2 Set CUR_DIR value

```
CUR_DIR=c:\HQProcess
```

WMQv7_HOME - Set this value to the WebSphere MQ installed directory

Example 15-3 Set WMQv7_HOME value

```
WMQv7_HOME= C:\Program Files\IBM\WebSphere MQ
```

HQQUEMGRNAME: The headquarters queue manager to which the headquarters application should connect

Example 15-4 Queue manager

```
HQQUEMGRNAME=QMHQ
```

HQPORT: The port number on which the headquarter queuemanager listens

Example 15-5 Port

```
HQPORT=1414
```

HQHOSTNAME: The host name on which the headquarters queue manager resides

Example 15-6 Host name

```
HQHOSTNAME=shmq.in.ibm.com OR
HQHOSTNAME=12.87.234.65
```

7. At the command prompt in the C:\HQProcess directory run the HQSetup.bat.

   ```
   C:\HQProcess.HQsetup.bat_
   ```

   Note: hqsetup.bat file internally uses the WebSphere MQ JMSAdmin tool to create the required Java Naming and Directory Interface objects. Ensure the JMSAdmin.config under the <WMQ Home>/java/bin directory is configured correctly, for more information on configuring the JMSAdmin.config file, refer section WMQ v7.0 Using Java manual -> Using the WebSphere MQ JMS administration tool.

15.3 Running the headquarters process

Follow the below steps to invoke the Store ordering application:

1. At the C:\HQProcess prompt, execute HQRun.bat passing the Initial Context Factory and Provider URL in the example below:

Example 15-7 `Execute HQRun.bat`

```
HQRun.bat com.sun.jndi.fscontext.RefFSContextFactory
file:/C:/Projects/MQSeries/JNDI-Directory
```

```
OR

java com.ibm.residency.HQOrderProcessing
-icf=com.sun.jndi.fscontext.RefFSContextFactory
-url=file:/C:/Projects/MQSeries/JNDI-Directory
```

Note: Enter the same value for -icf and -url as you would specify in the JMSAdmin.config file under the <WMQ Home>/java/bin directory.

2. Once the application starts the headquarters application is ready to process the requests from the stores.

15.4 Deploying the Store ordering component

Before the Store ordering application could be started, the following scripts must be run on the client machine.

1. Ensure WebSphere MQ client is installed on the machine where the Store Ordering application runs.
2. Ensure Java Development Kit 1.4.2 or higher is installed and configured on the machine.
3. The classpath variable must be set to include all the WebSphere MQ V7 jar files found under the <WMQ Home>/java/lib directory:

Example 15-8 Include WebSphere MQ V7 jar files

```
set classpath=.;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\CL3Export.zip;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\CL3Nonexport.zip;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\com.ibm.mqbind.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\connector.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\fscontext.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\jndi.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\jta.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\ldap.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\mqjbdf02.dll;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\mqjbnd05.dll;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\MQXAi02.dll;C:\Program Files\IBM\WebSphere MQ
```

```
v7.0\Java\lib\providerutil.jar;C:\Program Files\IBM\WebSphere MQ
v7.0\Java\lib\rmm.jar;
```

4. Extract the StoreOrdering.zip file to the c:\StoreOrdering directory

> **Note:** StoreOrdering.zip - contains all the files required for the Store ordering application to function.
>
> storeordering.jar – The main jar file that contains the class files.
>
> SOSetup.bat – Initial setup file for the Store ordering application.

5. Modify the classpath to include the storeordering.jar reference at the start of the classpath as shown below

   `set classpath=c:\StoreOrdering\storeordering.jar;%classpath%;`

6. Edit the sosetup.bat file and modify the following values

   `CUR_DIR - Set this value to the current working directory`

Example 15-9 Set CUR_DIR value

```
CUR_DIR=c:\StoreOrdering
```

WMQv7_HOME: Set this value to the WebSphere MQ installed directory

Example 15-10 Set WMQ_HOME value

```
WMQv7_HOME= C:\Program Files\IBM\WebSphere MQ
```

SOQUEMGRNAME: The headquarters queue manager to which the Store ordering application should connect

Example 15-11 Queue manager

```
SOQUEMGRNAME=QMHQ
```

SOPORT: The port number on which the head quarter queuemanager listens

Example 15-12 Port

```
SOPORT=1414
```

SOHOSTNAME: The host name on which the headquarters queue manager resides

Example 15-13 Host name

```
SOHOSTNAME=shmq.in.ibm.com OR
```

```
SOHOSTNAME=12.87.234.65
```

7. At the command prompt in the c:\StoreOrdering directory run the SOSetup.bat.:

   ```
   C:\StoreOrdering>SOSetup.bat
   ```

   Note: sosetup.bat file internally uses the WebSphere MQ JMSAdmin tool to create the required Java Naming and Directory Interface objects. Ensure the JMSAdmin.config under the <WMQ Home>/java/bin directory is configured correctly, for more information on configuring the JMSAdmin.config file, refer section WMQ v7.0 Using Java manual -> Using the WebSphere MQ JMS administration tool.

15.5 Invoking the Store ordering component

Follow the below steps to invoke the Store ordering application:

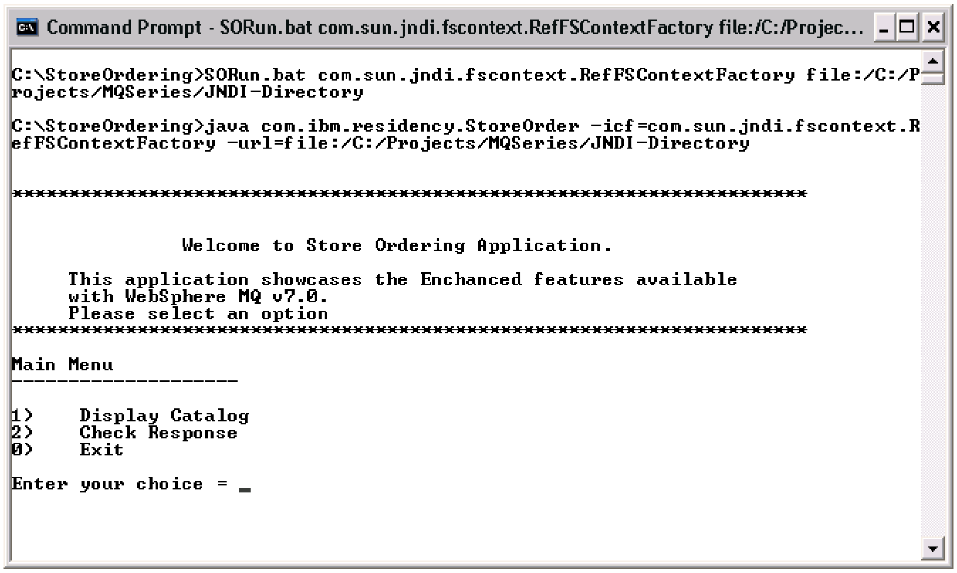
1. At the c:\StoreOrdering prompt, execute SORun.bat passing the Initial Context Factory and Provider URL as shown in the example below:

Example 15-14 Execute SORun.bat

```
SORun.bat com.sun.jndi.fscontext.RefFSContextFactory
file:/C:/Projects/MQSeries/JNDI-Directory

OR

java com.ibm.residency.StoreOrder
-icf=com.sun.jndi.fscontext.RefFSContextFactory
-url=file:/C:/Projects/MQSeries/JNDI-Directory
```

Note: Enter the same value for -icf and -url as you would specify in the JMSAdmin.config file under the <WMQ Home>/java/bin directory.

2. Once the application starts the user is presented with the following screen.



Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
C:\StoreOrdering>SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/P
rojects/MQSeries/JNDI-Directory

C:\StoreOrdering>java com.ibm.residency.StoreOrder -icf=com.sun.jndi.fscontext.R
efFSContextFactory -url=file:/C:/Projects/MQSeries/JNDI-Directory


**********************************************************************


               Welcome to Store Ordering Application.

     This application showcases the Enchanced features available
     with WebSphere MQ v7.0.
     Please select an option
**********************************************************************

Main Menu
--------------------

1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice  = _
```

Figure 15-2 Store order application

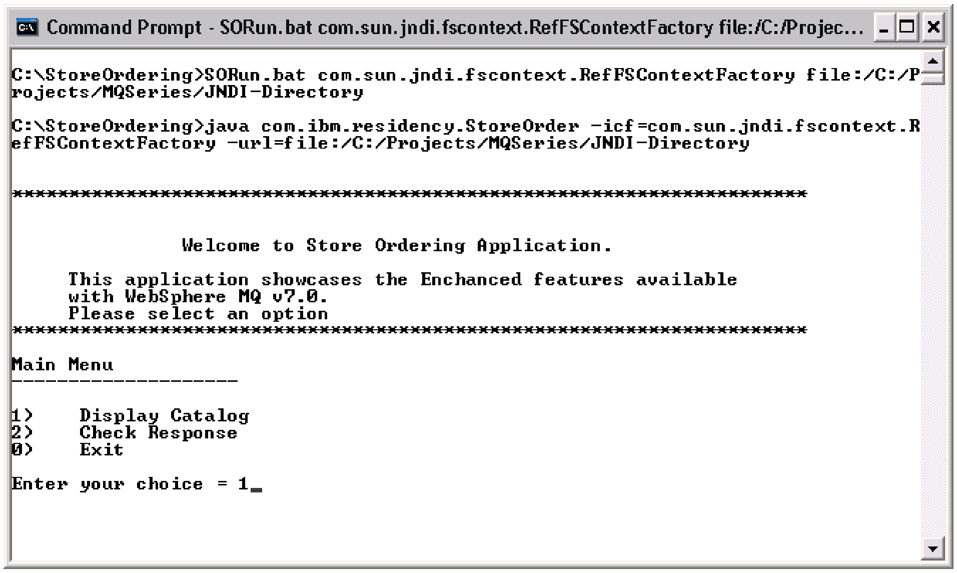
15.6 Running the Store ordering component

1. When the Store ordering application is invoked the user is presented with the options as shown in Fig 15-5

   Display Catalog: User can use this option to view the product catalog and then submit orders to the headquarters

   Check Response: User can use this option to view the responses for the previously submitted orders.

   Exit - Termminate the application.

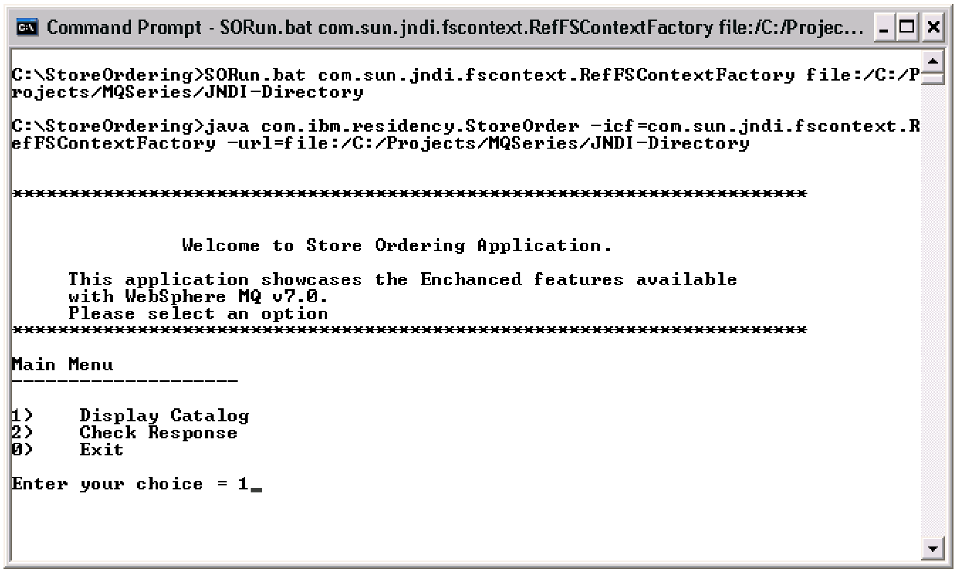


```
Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

C:\StoreOrdering>SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/P
rojects/MQSeries/JNDI-Directory

C:\StoreOrdering>java com.ibm.residency.StoreOrder -icf=com.sun.jndi.fscontext.R
efFSContextFactory -url=file:/C:/Projects/MQSeries/JNDI-Directory


**********************************************************************


               Welcome to Store Ordering Application.

     This application showcases the Enchanced features available
     with WebSphere MQ v7.0.
     Please select an option
**********************************************************************

Main Menu
--------------------

1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice = 1_
```

Figure 15-3

2. Displaying the product catalogue:

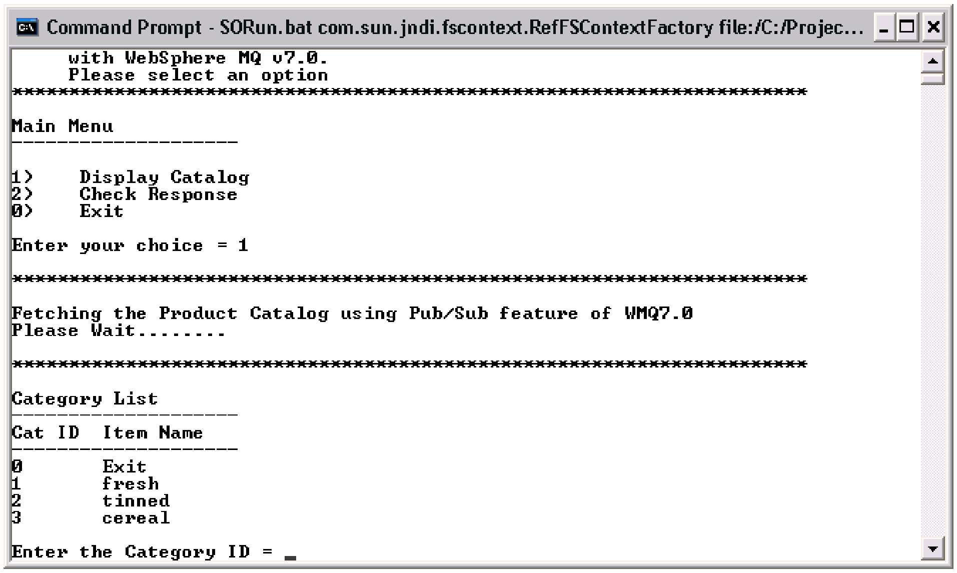
   Type 1 and press enter



```
Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

C:\StoreOrdering>SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/P
rojects/MQSeries/JNDI-Directory

C:\StoreOrdering>java com.ibm.residency.StoreOrder -icf=com.sun.jndi.fscontext.R
efFSContextFactory -url=file:/C:/Projects/MQSeries/JNDI-Directory


**********************************************************************


               Welcome to Store Ordering Application.

     This application showcases the Enchanced features available
     with WebSphere MQ v7.0.
     Please select an option
**********************************************************************

Main Menu
--------------------

1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice = 1_
```

Figure 15-4 Display product catalogue

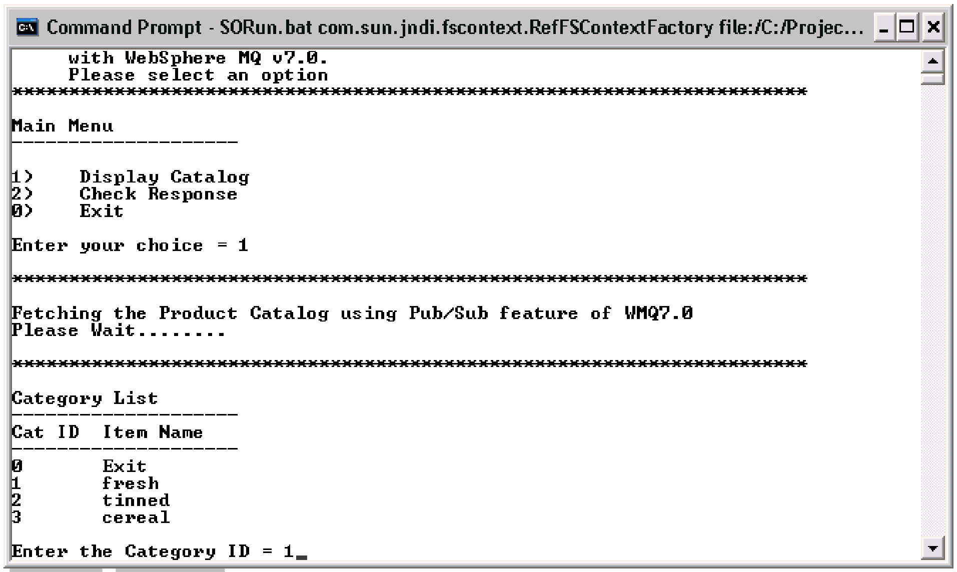
3. Once the user types 1 and presses enter, the user is presented with a list of product categories


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
     with WebSphere MQ v7.0.
     Please select an option
**********************************************************************

Main Menu
--------------------

1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice = 1

**********************************************************************

Fetching the Product Catalog using Pub/Sub feature of WMQ7.0
Please Wait........

**********************************************************************

Category List
--------------------
Cat ID  Item Name
--------------------
0       Exit
1       fresh
2       tinned
3       cereal

Enter the Category ID = _
```

Figure 15-5 Product catagories

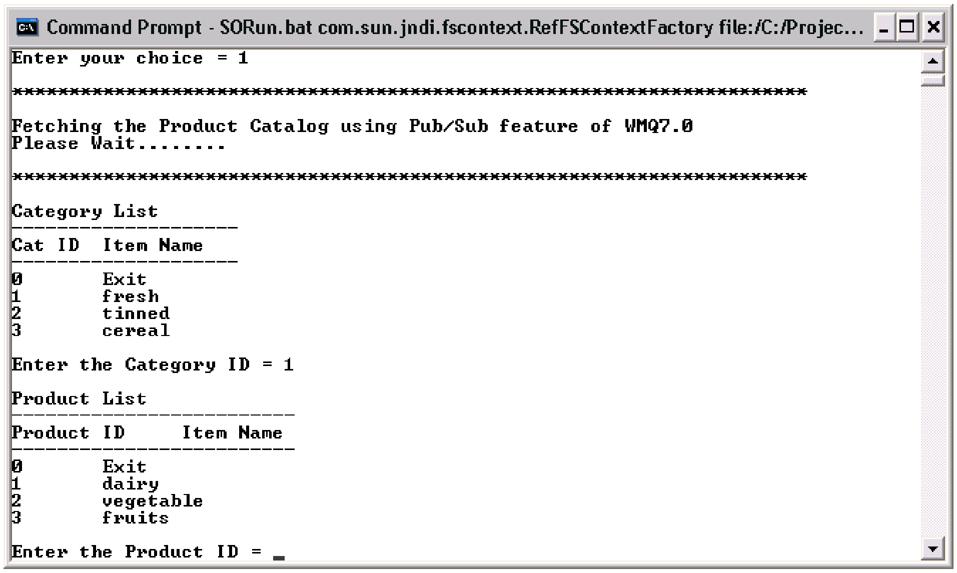
4. The user can select any of the product category displayed to place an order. To select category fresh from the list, Type 1 and press enter


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

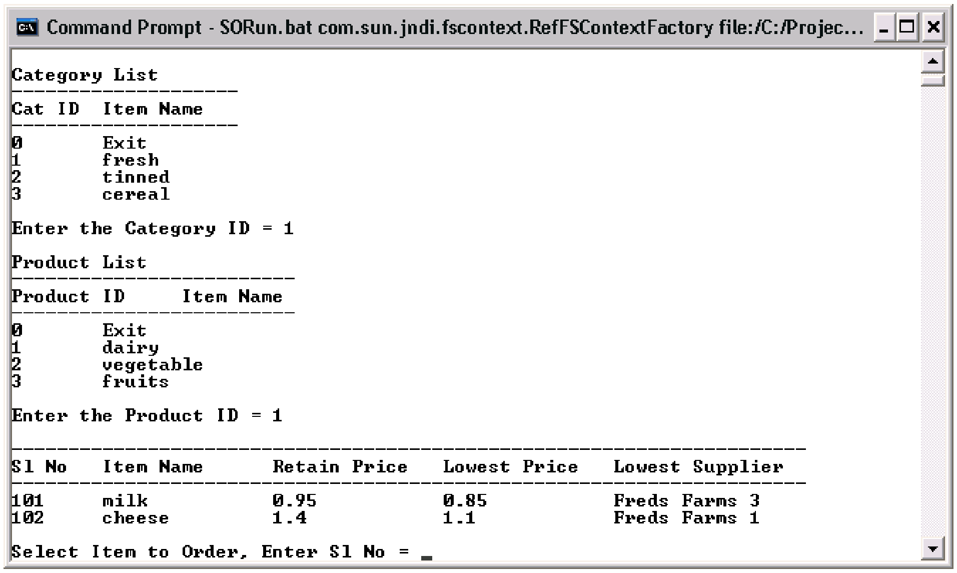
```
     with WebSphere MQ v7.0.
     Please select an option
**********************************************************************

Main Menu
--------------------

1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice = 1

**********************************************************************

Fetching the Product Catalog using Pub/Sub feature of WMQ7.0
Please Wait........

**********************************************************************

Category List
--------------------
Cat ID  Item Name
--------------------
0       Exit
1       fresh
2       tinned
3       cereal

Enter the Category ID = 1_
```

Figure 15-6 Select catagories

The user is provided with a list of Product types to choose from as shown in the Figure 15.9


```
Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...
Enter your choice = 1

**********************************************************************

Fetching the Product Catalog using Pub/Sub feature of WMQ7.0
Please Wait........

**********************************************************************

Category List
--------------------
Cat ID  Item Name
--------------------
0       Exit
1       fresh
2       tinned
3       cereal

Enter the Category ID = 1

Product List
-------------------------
Product ID     Item Name
-------------------------
0       Exit
1       dairy
2       vegetable
3       fruits

Enter the Product ID = _
```
Figure 15-7 Product types

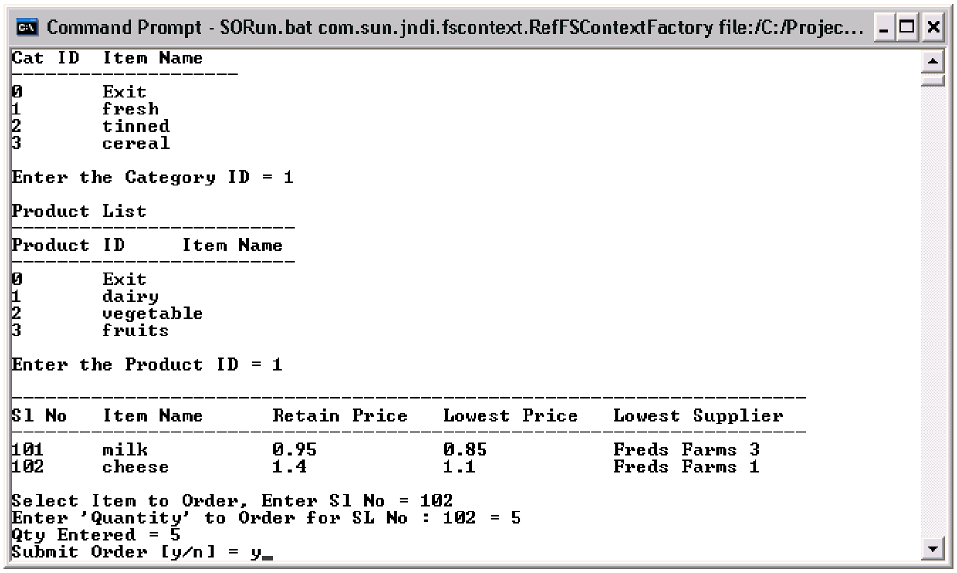
5. To view the different products under the product type, type the corresponding number. To view products under the dairy type, type 1 and press enter, the user is presented with the following products.


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
Category List
--------------------
Cat ID  Item Name
--------------------
0       Exit
1       fresh
2       tinned
3       cereal

Enter the Category ID = 1

Product List
-------------------------
Product ID      Item Name
-------------------------
0       Exit
1       dairy
2       vegetable
3       fruits

Enter the Product ID = 1

-----------------------------------------------------------------------
Sl No   Item Name       Retain Price    Lowest Price    Lowest Supplier
-----------------------------------------------------------------------
101     milk            0.95            0.85            Freds Farms 3
102     cheese          1.4             1.1             Freds Farms 1

Select Item to Order, Enter Sl No = _
```

Figure 15-8 Select product types

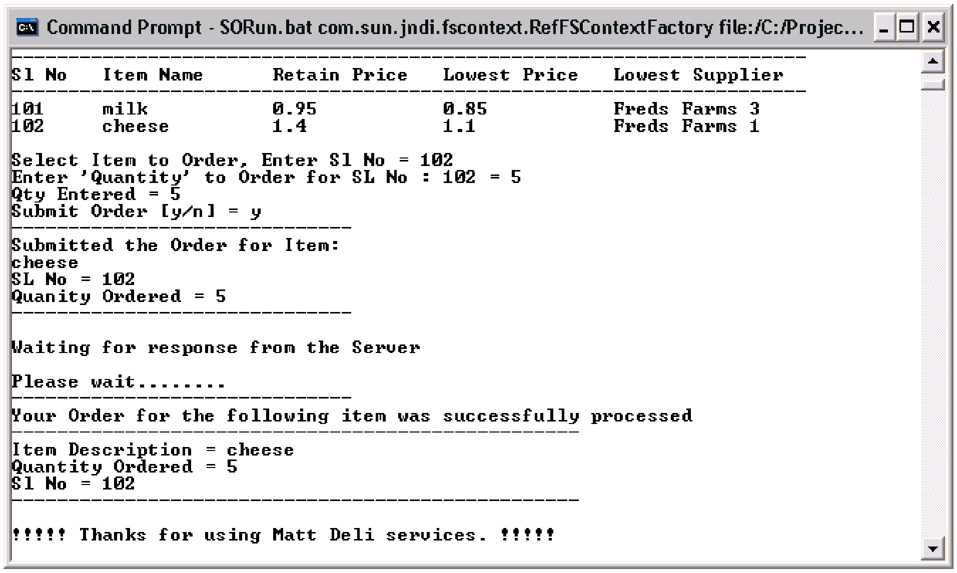
6. The user can now place the order for a product. Type the Sl No, Quantity and Submit order options as shown in the following figure.


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
Cat ID  Item Name
--------------------
0       Exit
1       fresh
2       tinned
3       cereal

Enter the Category ID = 1

Product List
-------------------------
Product ID      Item Name
-------------------------
0       Exit
1       dairy
2       vegetable
3       fruits

Enter the Product ID = 1

-----------------------------------------------------------------------
Sl No   Item Name       Retain Price    Lowest Price    Lowest Supplier
-----------------------------------------------------------------------
101     milk            0.95            0.85            Freds Farms 3
102     cheese          1.4             1.1             Freds Farms 1

Select Item to Order, Enter Sl No = 102
Enter 'Quantity' to Order for SL No : 102 = 5
Qty Entered = 5
Submit Order [y/n] = y_
```

Figure 15-9 Submit order

7. Press **Enter** key after entering the details. Once the order is submitted to the headquarters, the headquarters sends a response to the Stores indicating if the order was sucessfull or not as shown below:

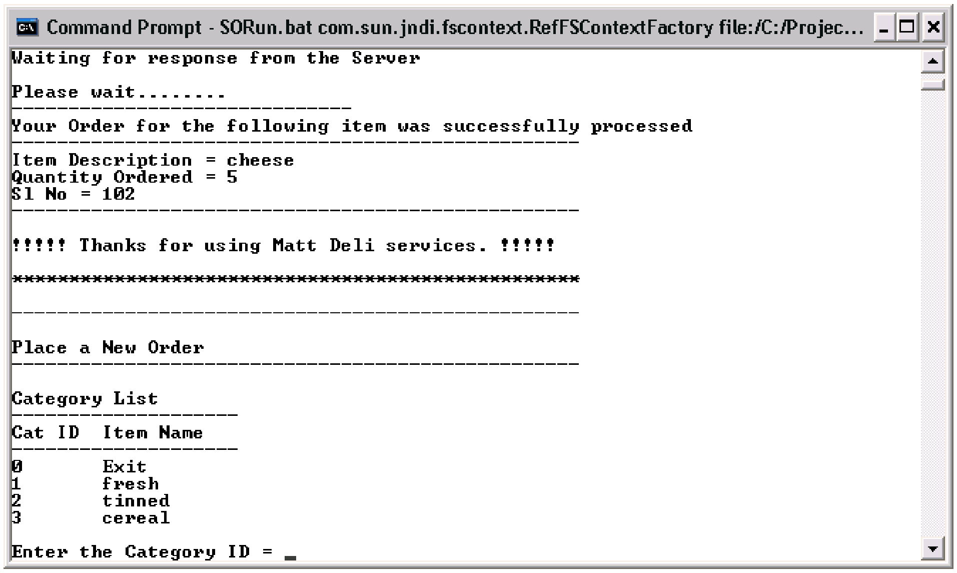


```
Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...
----------------------------------------------------------------------
Sl No   Item Name      Retain Price   Lowest Price   Lowest Supplier
----------------------------------------------------------------------
101     milk           0.95           0.85           Freds Farms 3
102     cheese         1.4            1.1            Freds Farms 1

Select Item to Order, Enter Sl No = 102
Enter 'Quantity' to Order for SL No : 102 = 5
Qty Entered = 5
Submit Order [y/n] = y
------------------------------
Submitted the Order for Item:
cheese
SL No = 102
Quanity Ordered = 5
------------------------------

Waiting for response from the Server

Please wait........
------------------------------
Your Order for the following item was successfully processed
--------------------------------------------------
Item Description = cheese
Quantity Ordered = 5
Sl No = 102
--------------------------------------------------

!!!!! Thanks for using Matt Deli services. !!!!!
```

Figure 15-10 Order response

Note: Fig 15-12 shows that the order for the product “cheese” was submitted by the stores to the headquarters. The headquarters then places this order to the Warehouse and sends the response to the Store ordering application.

8. The window displayed displays the product categories to allow the user to place a new order:



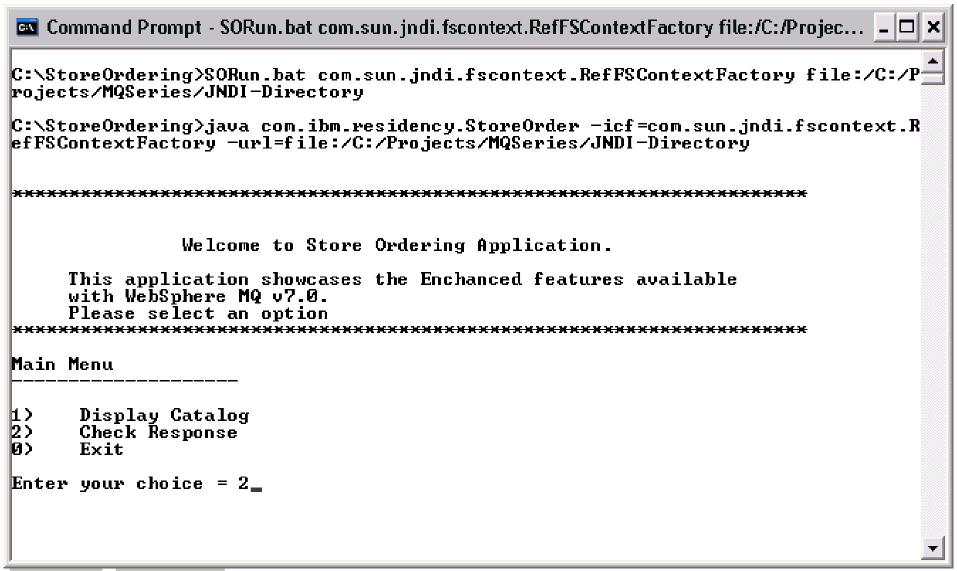
Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
Waiting for response from the Server

Please wait........
------------------------------
Your Order for the following item was successfully processed
--------------------------------------------------
Item Description = cheese
Quantity Ordered = 5
Sl No = 102
--------------------------------------------------

!!!!! Thanks for using Matt Deli services. !!!!!

**************************************************

--------------------------------------------------

Place a New Order
--------------------------------------------------

Category List
--------------------
Cat ID  Item Name
--------------------
0       Exit
1       fresh
2       tinned
3       cereal

Enter the Category ID = _
```

Figure 15-11 Place a new order

9. Check Responses

   In case the headquarters is not able to send the response to the Stores within 3 seconds the user is able to still check for the response asynchronously. On the main menu select the Check Reponses option as shown below:

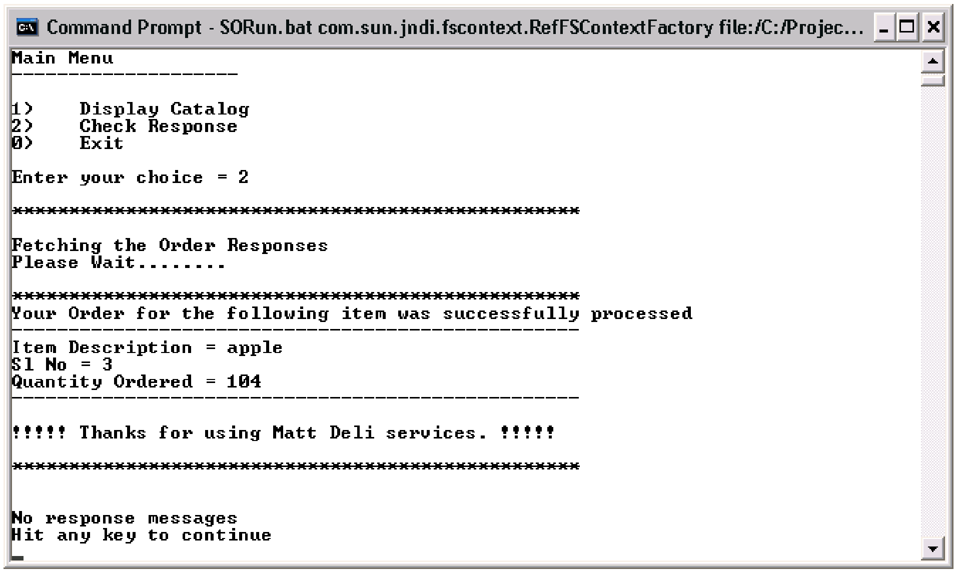


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```

C:\StoreOrdering>SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/P
rojects/MQSeries/JNDI-Directory

C:\StoreOrdering>java com.ibm.residency.StoreOrder -icf=com.sun.jndi.fscontext.R
efFSContextFactory -url=file:/C:/Projects/MQSeries/JNDI-Directory


**********************************************************************


               Welcome to Store Ordering Application.

     This application showcases the Enchanced features available
     with WebSphere MQ v7.0.
     Please select an option
**********************************************************************

Main Menu
--------------------


1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice = 2_
```

Figure 15-12 Select Check Response

10. The Store ordering application then displays the responses for the previous orders submitted

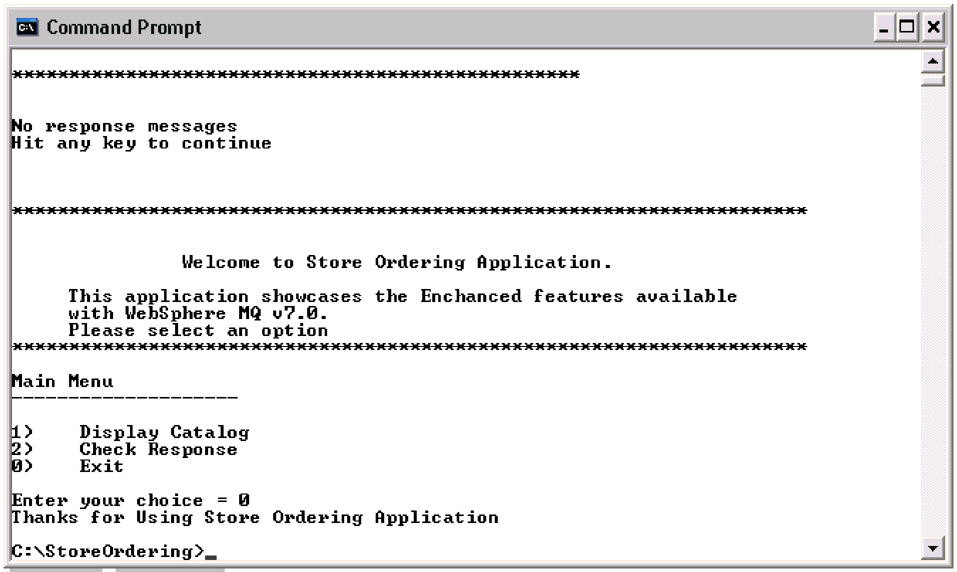


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
Main Menu
--------------------

1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice = 2

**************************************************

Fetching the Order Responses
Please Wait........

**************************************************
Your Order for the following item was successfully processed
--------------------------------------------------
Item Description = apple
Sl No = 3
Quantity Ordered = 104
--------------------------------------------------

!!!!! Thanks for using Matt Deli services. !!!!!

**************************************************


No response messages
Hit any key to continue
_
```

Figure 15-13 Order status

11. The user can type 0 and press enter at any time to go back to the main menu and to exit from the application.



Command Prompt

```
**************************************************


No response messages
Hit any key to continue



**********************************************************************


               Welcome to Store Ordering Application.

     This application showcases the Enchanced features available
     with WebSphere MQ v7.0.
     Please select an option
**********************************************************************

Main Menu
--------------------

1)     Display Catalog
2)     Check Response
0)     Exit

Enter your choice = 0
Thanks for Using Store Ordering Application

C:\StoreOrdering>_
```

Figure 15-14 Exit

15.7 Summary

This scenario demonstrates the new features of WebSphere MQ V7.0 enhanced for Java Message Service. The store ordering application uses the wild card '+' to subscribe. This allows the application to fetch only the immediate child topics. Also, the JMS application fetches the retained pulications stored at the WebSphere MQ on the headquarters. The connection factory objects can be changed to make use of the Asyhncronous put and the Read ahead mechanisms.

16

Scenario: News using Client

This component of the scenario illustrates two of the enhancements in WebSphere MQ V7.0 Client, asynchronous put and read ahead. These features are described in detail in Chapter 5., “WebSphere MQ Client enhancements” on page 61 of this book.

The new Publish/Subscribe feature of WebSphere MQ V7.0 is also used. Topic objects are defined to restrict access to topic strings. Refer to Chapter 4., “Publish/Subscribe integration” on page 45 of this book for details of Publish/Subscribe and the new topic object.

This chapter consists of the following sections:

- “Design overview”
- “Deploying the News component”
- “Running the News component”
- “Verifying the News component”
- “Summary”

16.1 Design overview

News items are generated at the headquarters of Matt's Deli and they are available to be viewed by staff the stores and the general public. This can be news like internal staff bulletins, announcements of special offers or new products. Each news item is stored in a separate MQ message as plain text.

Figure 16-1 shows the overall design. The programs and data are independent of all other components of the scenario. Refer to the figure when reading the following description.

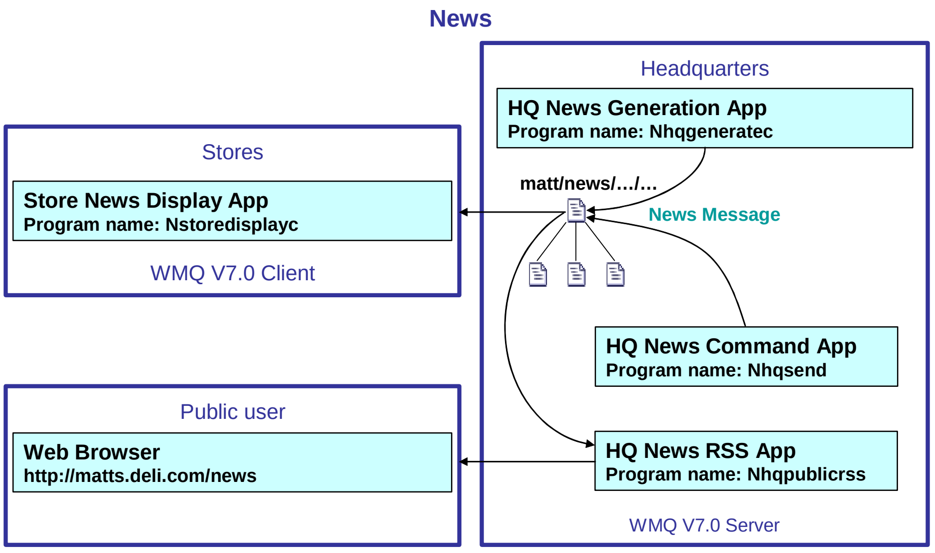


Figure 16-1 Programs and Data Flows in the News component

News items are published to two layers of topics beneath the tree *matt/news* on the headquarters queue manager using two different programs which are run by headquarters staff. One of these is designed to read news from a text file which has been prepared beforehand. The other program allows individual news items to be manually entered one by one as needed.

At the stores, a program is started by a staff member every morning to subscribe to all news topics. The news items are displayed on an overhead monitor as they are published.

The public can also run this program to connect to headquarters and subscribe to a sub-set of the news topics, but in this scenario they use a RSS feed (Really

Simple Syndication). A web browser with an appropriate RSS reader plug-in can view the RSS feed, or a dedicated RSS reader program can be used.

There is a Java script program which runs continuously at headquarters. It subscribes to all public news topics and accumulates them in an Extensible Markup Language (XML) file in RSS compliant format.

16.2 Deploying the News component

Some of the programs are designed run on the same system as the headquarters queue manager. The programs at the stores and the public web browser or RSS reader can be run on other systems, but they can also be run on the headquarters system to simplify the deployment.

Executable programs are supplied for the Windows platform. C source and make files are also included so that they can be recompiled. This is especially useful if the programs are to be run on Unix or other platforms which support C.

Listings of the data files and Windows command files are provided when they are used in 16.4, “Verifying the News component” on page 310.

16.2.1 Copying files

The file `Chp 16 News.zip` is supplied with the materials in Appendix B, “Additional material” on page 343. To deploy the news component, extract the files listed in the following table from the zip file into the `C:\Scenario\News` folder on the headquarters and stores systems:

Table 16-1 News component files

Program	File name(s)	Description
News generation	Nhqgeneratec.exe	Windows executable program which runs at headquarters. It uses MQ Client to read data records from standard input and it publishes the news items to the specified topics. It demonstrates the new “asynchronous put” feature for improved performance compared with previous versions of MQ.
	testgendata1.txt	Sample data for generating news items on topics.

Program	File name(s)	Description
	runnewsgendelay.bat runnewsgenfast.bat	Command files to run the Nhqgeneratec program with specific parameters
News command	Nhqsend.exe	Windows executable program which runs at headquarters. It read data records from standard input and publish the news items to a specific news topic. It demonstrates publishing to a topic using the MQPUT1 verb.
	runnewssend.bat	Command file to run the Nhqsend program with specific parameters.
News RSS generation	Nhqpublicrss.java	Java program which runs at headquarters. It subscribes to the public news topics and generates a RSS XML file.
	runrssgenxml.bat	Command file to run the Nhqnewsgetter program.
Store news display	Nstoredisplayc.exe	Windows executable program which runs at the stores. It uses MQ Client to subscribe to news topics and display the news items. It demonstrates subscribing and also the new “read ahead” feature for improved performance compared with previous versions of MQ.
	runstoredisplay.bat	Command file to run the Nstoredisplayc program with specific parameters.

16.2.2 Public web browser

A web browser, such as Microsoft Internet Explorer or Mozilla Firefox, with the appropriate RSS reader plug-in installed, can be used on any supported platform to open and display a RSS feed of Matt’s Deli public news items. These are not supplied with the materials in Appendix B, “Additional material” on page 343.

16.2.3 Compiling the Windows executable programs

The C source files for all the programs are provided in the file `Chp 16 News.zip`. A make file and batch file are also included to generate the executable programs on Windows using a Microsoft command line compiler and linker, such as the free *Microsoft Visual C++ 2005 Express Edition*. It requires WebSphere MQ V7.0 Server to be installed as the programs call the new verbs in the MQ API. The compiler is not supplied with the materials in Appendix B, “Additional material” on page 343.

The make file needs to be changed to allow the programs to be compiled and linked on Unix. Details of the required changes are not covered in this book because it is heavily dependent on the type of Unix platform. Refer to the manual *WebSphere MQ Application Programming Guide* SCXX-XXXX-XX for information on compiling the programs on the supported Unix platforms.

The files are listed in the following table:

Table 16-2 Files used to compile the news component

File name(s)	Description
Nhqgenerate.c Nhqsend.c Nstoredisplay.c	The C source files for all the executable programs. They are independent and do not require any other header files or libraries, except the standard Windows and MQ libraries.
makenews.txt	A “make” file for the C programs on Windows. This would normally be named “makefile” on Unix systems and requires modification to use the Unix compiler and linker.
make.bat	A command file which uses makenews.txt to compile and link the C programs on Windows.

16.2.4 MQ Objects

The news component uses the queue manager at headquarter and does not require any queue objects to be defined, it is purely Publish/Subscribe and lets MQ manage the subscription queues. Two topic objects are defined to allow OAM security profiles to restrict access to the news topic strings. News requires two SVRCONN channels objects to be defined so that MQ Client programs can connect to the queue manager from remote systems and the headquarters system.

The following MQSC commands are included in `QMHQobjects.txt`, which is run during the scenario preparation steps described in Chapter 13., “Scenario preparation” on page 251:

Example 16-1 MQSC commands for the news component

```
DEFINE CHANNEL(QMHQ_STORES) +
   CHLTYPE(SVRCONN) +
   TRPTYPE(TCP) +
   REPLACE
DEFINE CHANNEL(QMHQ_NEWS) +
   CHLTYPE(SVRCONN) +
   TRPTYPE(TCP) +
   REPLACE
DEFINE TOPIC(MATT.TOPIC.NEWS.INTERNAL) +
   TOPICSTR('matt/news/internal') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE
DEFINE TOPIC(MATT.TOPIC.NEWS.PUBLIC) +
   TOPICSTR('matt/news/public') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE
```

The following Object Authority Manager (OAM) commands are included in `QMHQsetaut.bat`, which is also run during the scenario preparation:

Example 16-2 OAM commands for the news component

```
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.* -g hq -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.* -g hq +pub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.INTERNAL -g st -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.INTERNAL -g st +sub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.PUBLIC -g matt -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.PUBLIC -g matt +sub
```

This allows users in the *hq* group at headquarters to publish to all news topics, and allows users in the *st* and *matt* groups to subscribe to all news topics.

16.3 Running the News component

In a Publish/Subscribe design using non-durable subscriptions and non-retained publications, it is normal to start the subscribers first and then run the publishers.

Non-durable subscribers don't care if not all published messages are received. Non-retained published messages are discarded if there are no durable subscriptions and there are no non-durable subscribers running at the time of publication.

This section provides general information on how to run the programs, including details of the environment and command line arguments which they require.

16.3.1 Starting the news display program at the stores

`Nstoredisplayc.exe` requires the MQSERVER environment variable to be set to specify the client channel name, the transport protocol, and the host and port number of the headquarters queue manager.

Open a Windows command prompt window and enter the following command. The host *matts.deli.com* and port number *1414* needs to be changed to whatever is actually being used by the TCP/IP listener for the headquarters queue manager:

```
set MQSERVER=QMHQ_STORES/TCP/matts.deli.com(1414)
```

The program requires a news category to be specified on the command line. This corresponds to a topic string to which the program subscribes, as per the following table:

Table 16-3 News categories for store display program

News category on command line	Program subscribes to topic string
internal	matt/news/internal/#
public	matt/news/public/#
all	matt/news/#

For example, the program can be run to only display internal news, for viewing by the stores:

```
Nstoredisplayc internal
```

16.3.2 Starting the news RSS generation program at headquarters

Nhqpublicrss runs at headquarters. It is a compiled Java class which subscribes to the public news topics and generates a RSS XML file. It requires the CLASSPATH environment variable to be set to include the required Java Archive (jar) files and the PATH environment variable set to include the 'bin' folder for the Java runtime.

Start the program using the following command:

```
java Nhqpublicrss.class
```

16.3.3 Starting the public news RSS reader

Open the following URL in a web browser which has a RSS plug-in, or open it in a special purpose RSS Reader program:

http://matts.deli.com/news.xml

matts.deli.com needs to be changed to the host name of the headquarters system. It displays the public news items.

16.3.4 Running the news generation program at headquarters

Nhqgeneratec.exe requires the MQSERVER environment variable to be set to specify the client channel name, the transport protocol, and the host and port number of the headquarters queue manager.

Open a Windows command prompt window and enter the following command. The host *matts.deli.com* and port number *1414* needs to be changed to whatever is actually being used by the TCP/IP listener for the headquarters queue manager:

```
set MQSERVER=QMHQ_NEWS/TCP/matts.deli.com(1414)
```

Some test data is supplied on file *testgendata1.txt*. The program can be run to publish each record of the test data as fast as possible:

```
Nhqgenerate -fast <testgendata1.txt
```

This provides good throughput using the asynchronous put feature. The file consists of the following data records, which consist of two comma separated fields containing the topic name and the news item.

Example 16-3 Contents of file testgendata1.txt

```
matt/news/public/newproduct,New season oranges are now available at all stores
matt/news/public/offers,Buy 3 lemons and get 1 free
matt/news/public/general,Matt's Deli is opening a new store in Chandlers Ford
next month
matt/news/internal/staff,Celebrate Matt's birthday with him at The Dolphin on
Friday night
matt/news/internal/products,Ensure you have plenty of lemons to keep up with
demand for the current special offer
```

```
matt/news/public/newproduct,We are proud to announce that imported cackle
berries are now available
matt/news/public/offers,Salami can now be purchased in ultra thin slices
matt/news/public/general,Our Otterbourne store is open to 10PM on Friday nights
matt/news/internal/staff,Don't forget that next Monday is a public holiday
matt/news/internal/products,Please check all stocks of yoghurt to made sure its
not past the use-by-date
matt/news/public/newproduct,We now have a range of cold-climate olives from
Greenland
matt/news/public/offers,Fresh bacon will be sold for half price on Saturday
morning only
matt/news/public/general,Santa is appearing at the Eastleigh store on Sunday
from 9AM to 11AM
matt/news/internal/staff,Cardboard boxes will be collected for recycling over
the weekend
matt/news/internal/products,Tinned goods deliveries from the warehouse will now
be on Wednesdays
```

The program can also be run to delay the publishing of each news item and also to repeat the set of news items a specified number of times. This does not make good use of the asynchronous put feature, but it does simulate a regular feed of news items which can be subscribed and displayed by the other programs:

```
Nhqgeneratec -delay 5 -repeat 100 <testgendata1.txt
```

This delays 5 seconds between each published news item and repeats all the news items in the data file 100 times before the program terminates.

16.3.5 Running the news command line program at headquarters

Nhqsend.exe runs on the headquarters system and uses a direct server binding to the local queue manager, so it does not require a client channel to operate.

The program command line argument indicates the topic string to which the program publishes news items:

Table 16-4 News categories for headquarters send program

News topic on command line	Program publishes to topic string
public/newproduct	matt/news/public/newproduct
public/offers	matt/news/public/offers
public/general	matt/news/public/general
internal/staff	matt/news/internal/staff

News topic on command line	Program publishes to topic string
internal/products	matt/news/internal/products

Open a Windows command prompt window and enter the following command:

```
Nhqsend internal/products
```

The program prompts for standard input. Enter a news item on each line and it is immediately published to the topic string. Enter a blank line to terminate the program.

16.4 Verifying the News component

This section provides two test cases which verify the correct operation of the news component. It gives an insight into how the programs interact and use the new features.

Command files are run in a specific order and the expected output from the programs is shown. The command files which run client programs need to be modified to replace the host name *matts.deli.com* and the port number *1414* to whatever is actually being used.

16.4.1 News is generated at headquarters and displayed by the stores

On a store system, start the program to subscribe and display all news topics by double clicking on **runstoredisplay.bat** in Windows Explorer. The file contains the following commands:

```
set MQSERVER=QMHQ_STORES/TCP/matts.deli.com(1414)
Nstoredisplayc all
pause
```

It displays news items for all the internal and public topics. The command prompt window should display the following output:

Example 16-4 Initial output of news display program at store

```
C:\Scenario\News>set MQSERVER=QMHQ_STORES/TCP/matts.deli.com(1414)
C:\Scenario\News>Nstoredisplayc all
**********************************
*  Nstoredisplay program starts   *
```

```
**********************************
<*> Connecting to default Queue Manager
<*> Starting non-durable subscription to topic string 'matt/news/#'
```

This program provides good throughput using the read ahead feature when there is a high volume of news items.

On the headquarters system, start the program to quickly publish test data to all five news topics, by double clicking on **runnewsgenfast.bat** in Windows Explorer. The file contains the following commands:

```
set MQSERVER=QMHQ_NEWS/TCP/matts.deli.com(1414)
Nhqgeneratec -fast <testgendata1.txt
pause
```

The command prompt window should display the following output:

Example 16-5 Output of news generation program at headquarters

```
C:\Scenario\News>Nhqgenerate -fast <testgendata1.txt
********************************
*  Nhqgenerate program starts  *
********************************
<*> Reading records from standard input in format 'topic,data'
<*> 15 records read from standard input
<*> Connecting to default Queue Manager
<*> All 13 publications completed OK
******************************
*  Nhqgenerate program ends  *
******************************
C:\Scenario\News>pause
Press any key to continue . . .
```

The news display program at the store should immediately display the 13 published news items. The last few lines of output should appear as follows:

Example 16-6 Updated output of news display program at store

```
<*> News arrived at 2007-12-12 16:20:38
public/newproduct
We now have a range of cold-climate olives from Greenland

<*> News arrived at 2007-12-12 16:20:38
public/offers
Fresh bacon will be sold for half price on Saturday morning only
```

```
<*> News arrived at 2007-12-12 16:20:38
public/general
Santa is appearing at the Eastleigh store on Sunday from 9AM to 11AM

<*> News arrived at 2007-12-12 16:20:38
internal/staff
Cardboard boxes will be collected for recycling over the weekend

<*> News arrived at 2007-12-12 16:20:38
internal/products
Tinned goods deliveries from the warehouse will now be on Wednesdays
```

On the headquarters system, start the program to interactively publish news items to one specific topic by double clicking on **runnewssend.bat** in Windows Explorer. The file contains the following commands:

```
Nhqsend internal/staff
pause
```

The command prompt window should prompt for a news item to be entered. Enter some text. Enter **shutdown** and then enter a blank line to end the program. The output should be similar to the following. The user input is shown in bold.

Example 16-7 Output of news send program at headquarters

```
C:\Scenario\News>Nhqsend internal/staff
****************************
*  Nhqsend program starts  *
****************************
<*> Connecting to default Queue Manager
<*> Opening topic string 'matt/news/internal/staff'

Enter a news item for 'internal/staff' or a blank line to end the
program
you can all go home now
<*> Publishing news message 'matt/news/internal/staff,you can all go
home now'

Enter a news item for 'internal/staff' or a blank line to end the
program
shutdown
<*> Publishing news message 'matt/news/internal/staff,shutdown

Enter a news item for 'internal/staff' or a blank line to end the
program
```

```
blank line
**************************
*  Nhqsend program ends  *
**************************
C:\Scenario\News>pause
Press any key to continue . . .
```

Press any key to close the window. In the mean time, the news display program window should have progressively displayed the following:

Example 16-8 Updated output of news display program at store

```
<*> News arrived at 2007-12-12 16:46:02
internal/staff
you can all go home now

<*> News arrived at 2007-12-12 16:46:14
internal/staff
shutdown
********************************
*  Nstoredisplay program ends  *
********************************
C:\Scenario\News>pause
Press any key to continue . . .
```

Press any key to close the window. This concludes verification of news generation at headquarters and news display at the stores.

The next section runs the RSS generation program and verifies than a RSS reader can view the news items.

16.4.2 News is generated and displayed by a RSS reader

On the headquarters system, start the Java program to subscribe to all news topics and generate the RSS XML file. A Windows command file is provided to compile and run the program. The files. The file contains the following commands:

Example 16-9 Contents of runrssgenxml.bat

```
@echo off
echo CLASSPATH must contain at least the three WebSphere MQ jar files
echo "com.ibm.mq.jmqi.jar", "com.ibm.mqjms.jar" and "com.ibm.mq.jar".
echo These are normally in the folder "C:\Program Files\IBM\WebSphere
MQ\Java\lib".
```

```
echo To be safe, they are all added to CLASSPATH.

set CLASSPATH=%CLASSPATH%;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.jmqi.jar
set CLASSPATH=%CLASSPATH%;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar
set CLASSPATH=%CLASSPATH%;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mq.jar
echo CLASSPATH=%CLASSPATH%
pause

echo PATH must contain the "bin" folder for the installed Java SDK.
echo This example assumes the IBM Java 50 SDK has been installed.
echo The set command will need to be changed if another Java SDK is being used.
echo The current folder must also be added so that the java command can find
the
echo compiled class file.

set PATH=%PATH%;C:\Program Files\IBM\Java50\bin;.
path
pause

echo Compile the Java source file into a class file

del /q Nhqpublicrss.class
echo Command: javac Nhqpublicrss.java
javac Nhqpublicrss.java
pause

echo Run the class file

echo Command: java Nhqpublicrss.class
java Nhqpublicrss.class
pause
```

Double click on **`runrssgenxml.bat`** in Windows Explorer. Press any key when prompted. The command prompt window should intially display the following output:

Example: 0-1 Initial output of news RSS generation program

```
CLASSPATH must contain at least the three WebSphere MQ jar files
"com.ibm.mq.jmqi.jar", "com.ibm.mqjms.jar" and "com.ibm.mq.jar".
These are normally in the folder "C:\Program Files\IBM\WebSphere MQ\Java\lib".
To be safe, they are all added to CLASSPATH.
CLASSPATH=C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar;C:\Progra
m Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\WebSphere
```

```
MQ\Java\lib\com.ibm.jmqi.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.
mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.jar
Press any key to continue . . .
PATH must contain the "bin" folder for the installed Java SDK.
This example assumes the IBM Java 50 SDK has been installed.
The set command will need to be changed if another Java SDK is being used.
The current folder must also be added so that the java command can find the
compiled class file.
PATH=C:\Program Files\IBM\WebSphere
MQ\Java\lib;C:\WINDOWS\system32;C:\WINDOWS;C
:\WINDOWS\System32\Wbem;C:\Program Files\IBM\WebSphere MQ\bin;C:\Program
Files\I
BM\WebSphere MQ\tools\c\samples\bin;C:\Program Files\IBM\Java50\bin;.
Press any key to continue . . .
Compile the Java source file into a class file
Command: javac Nhqpublicrss.java
Note: Nhqpublicrss.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Press any key to continue . . .
Run the class file
Command: java Nhqpublicrss.class
Listening on topic topic://matt/news/public/general
Listening on topic topic://matt/news/public/newproduct
Listening on topic topic://matt/news/public/offers
Listening on topic topic://matt/news/public/general
Listening on topic topic://matt/news/public/newproduct
Listening on topic topic://matt/news/public/offers
```

Also on the headquarters system, start the program to slowly publish test data to all five news topics, by double clicking on **runnewsgendelay.bat** in Windows Explorer. The file contains the following commands:

```
set MQSERVER=QMHQ_NEWS/TCP/matts.deli.com(1414)
Nhqgeneratec -delay 2 -repeat 10 <testgendata1.txt
pause
```

The program reads all the records from standard input into memory and then publish news items with a 2 second delay between each item. It repeats the sequence of records 10 times and then terminate.

After about 15 seconds of running the command prompt window should display the following output:

Example 16-10 Initial output of news generation program at headquarters

```
C:\Scenario\News>set MQSERVER=QMHQ_NEWS/TCP/matts.deli.com(1414)
C:\Scenario\News>Nhqgeneratec -delay 5 -repeat 10 0<testgendata1.txt
```

```
********************************
*  Nhqgenerate program starts  *
********************************
<*> Reading records from standard input in format 'topic,data'
<*> 15 records read from standard input
<*> Connecting to default Queue Manager
<*> Publishing message:
  Topic 'matt/news/public/newproduct'
  Data  'matt/news/public/newproduct,New season oranges are now
available at all stores'
<*> Publishing message:
  Topic 'matt/news/public/offers'
  Data  'matt/news/public/offers,Buy 3 lemons and get 1 free'
<*> Publishing message:
  Topic 'matt/news/public/general'
  Data  'matt/news/public/general,Matt's Deli is opening a new store in
Chandlers Ford next month'
```

Open the following URL in a web browser which has a RSS plug-in, or open it in a special purpose RSS Reader program:

`http://matts.deli.com/news.xml`

matts.deli.com needs to be changed to the host name of the headquarters system. It displays the public news items.

Verify that the news items appear soon after they are published by the *Nhqgenerate* program. After about 10 minutes *Nhqgenerate* terminates and no new news items should appear in the RSS reader. The final output of the program is as follows:

Example 16-11 Final output of news generation program at headquarters

```
<*> Publishing message:
  Topic 'matt/news/internal/staff'
  Data  'matt/news/internal/staff,Cardboard boxes will be collected for
recycling over the weekend'
<*> Publishing message:
  Topic 'matt/news/internal/products'
  Data  'matt/news/internal/products,Tinned goods deliveries from the
warehouse will now be on Wednesdays'
******************************
*  Nhqgenerate program ends  *
******************************
C:\Scenario\News>pause
```

```
Press any key to continue . . .
```

16.5 Summary

The WebSphere MQ Client is used to implement a news distribution capability for Matt's Deli, where the users are remote from the central headquarters system. Two enhancements in MQ Client are demonstrated as being appropriate to the solution design. Non-durable subscriptions and non-retained publications are also used with a hierarchical topic tree of news categories.

The news generation program used the new asynchronous put feature to publish a large number of messages without waiting for the response on each MQPUT. The MQSTAT call at the end of the program confirmed that all of the MQPUTs succeeded.

The news display program in the stores used the new read ahead feature to get news subscription messages and demonstrate an improved throughput of batches of messages in V7.0 compared with previous versions of MQ.

17

Scenario: Web ordering over HTTP

17.1 Web ordering uses cases

*** Remember nothing at on end user machine ***

17.2 Deploying the scenario component

17.2.1 Java script sample

17.2.2 HTTP Bridge

17.2.3 Headquarter application code

Compiling the source

17.3 Running the scenario component

17.3.1 Start the HTTP Bridge

17.3.2 Invoking the application from the web page

17.3.3 Component verification

17.4 Summary

Note to Author: The introduction to the chapter should summarize what the chapter covers in a couple of sentences or paragraphs. Do not list out each section unless highlighting specific information to the reader. Describe the chapter contents here using these or similar words. Optionally add level 2 headings to a list using: **Special** → **Cross-Reference** → **Format: Head** → **Insert**

This chapter provides, describes, discusses, or contains...

In this chapter we introduce, provide, describe, or discuss...

In this chapter:
In this chapter, the following topics are discussed, described:
This chapter provides, describes, discusses, or contains the following:

- ...
- ...
- Sample level 2 “n.n” chapter heading
  (created by **Special** → **Cross-Reference** → **Format: Head** → **Insert**)
- Sample next level 2 heading

17.5 Sample level 2 “n.n” chapter heading (Head 1) new page

Note to Author: The first level 2 “n.n” heading in a chapter should be the (Head 1) tag, skip to new page.

Add text here (Body0).

17.6 Sample level 2 “n.n” chapter heading (Head 2)

Add text here (Body0).

17.6.1 Sample level 3 “n.n.n” chapter heading (Head 3)

Add text here (Body0).

Sample level 4 heading (Head 4)

Add text here (Body0).

Sample level 5 heading (Head 5)

Add text here (Body0)

Sample bulleted list

- ► Add text here (ListBulleted 1)
- ► Add text here (ListBulleted 1)

  Add text here at list 1 level (Body1)

  - – Add text here (ListBulleted 2)

    Add text here at list 2 level (Body2)

    - • Add list text here (ListBulleted 3)

      Add text here at list 3 level (Body3)

Sample numbered list

1. Add text here (ListNumber 1)
2. Add text here (ListNumber 1next)

Add text here at list 1 level (Body1)

a. Add text here (ListNumber 2)

   Add text here at list 2 level (Body2)

Note to Author: Display the **Paragraph Catalog,** click the **"¶"** in the upper right of an open FrameMaker file.

Sample Shaded Box

Issue: **Toolkit → RXFM → Insert_tools → Insert Shaded Box**

Note: You can also use the **Toolkit → Quick Access Menu** to issue the **Toolkit → RXFM → Insert_tool** commands

Sample Figure

Issue: **Toolkit → RXFM → Insert_tools → Insert Figure NoBox For Graphics**

Figure 17-1 Sample caption: to make this figure frame page wide, issue: ***Toolkit → RXFM → Author_tools → Paragraph-Object-Shift-Left***

Sample Table

Issue: **Toolkit → RXFM → Insert_tools → Insert Table 2x2**

Table 17-1 Sample caption

Sample Example

Issue: **Toolkit → RXFM → Insert_tools → Insert Example**

Example 17-1 Sample caption

Note to Author: **Character Catalog**
Display the **Character Catalog**, click the "**f**" in the upper right hand corner of an open FrameMaker file. Select (highlight) the text you want to change.

- **Default ¶ Font** - Use this tag to remove any of the highlight tags that you apply from this catalog and to return the text to the default format.
- **Citation** - Use this tag to italicize the title of publications. Select the text and click Citation. A title would look like this: *Framemaker and Writing Guidelines for Residents*, SG24-1234. Note that the number is not in italics.
- **Commands** - Use this tag for lowercase (that is, not VM or MVS) commands, such as UNIX® commands like `make`. Note: Use this Commands tag for inline commands, not for command statements that are on separate lines.
- **Emphasis** - Use this tag to *emphasize* (highlight) a word or phrase. To change your selection back to normal font, use Default ¶ Font. You can also use "double quotes" to highlight text.
- **Example -** Use this font to change a word or phrase within a paragraph of normal text to nonproportional font. (For entire paragraphs, use BodyExample0.) To change your selection back to normal font, use Default ¶ Font.
- **Bold** - Use this tag for actions selected from a menu, such as **Format** → **Document** → **Text Options**.
- **Italic** - Use this tag for variables in code specifications, as in ABC.123.*names*.

Note to Author: **Attributions**
Attributing material created or published by companies other than IBM. This third party material or open source material can include text, artwork, screenshots, software, excerpts from manuals and help text.
Important: Unless otherwise instructed, put the attribution into a footnote.
Use the standard footnote command:

1. Put the cursor where you want the footnote # in the FrameMaker source text.
2. Issue: **Special** → **Footnote**
3. Add footnote text as:

- For brief quotes of text, not requiring permission, use the footnote text, "*author, title, publisher, date, pages*." or "Web site(if applicable)"
- For un-modified text, screenshots, graphs, or tables, where permission is required and granted, use the footnote text, "*company-name*, Reprinted by Permission."
- For modified or reformatted material, where permission is required and granted, use the footnote text, "Source: *author, title, date*."

Details about attributions, "fair use", photos, Microsoft screenshots, and when to obtain written permissions, see the ITSO Teamroom item: "Attribution Guidelines for Project Leaders and Editors" or at:
`http://w3.itso.ibm.com/attributions`
Note: Trademark name attributions are handled by the Trademarking tools. The Trademark names do not have to be manually marked.

18

Scenario: Warehousing using call back

This component of the scenario illustrates the use of call back for asynchronous consume.

This scenario demonstrates how an application that is not programmed to participate in Publish/Subscribe as subscriber can receive publications.

The following topics are discussed:

- “Design overview”
- “Deploying the Warehousing component”
- “Running the Warehousing component”
- “Verifying the Warehousing component”
- “Summary”

18.1 Design overview

Matt’s Deli has one warehouse that receives deliveries from all suppliers and dispatches the products ordered to the stores. The warehouse management system that runs on a z/OS mainframe requires that copies of all purchase orders are sent to the warehouse. The warehouse prefers to receive the orders in batches at certain times during the day.

The warehouse mainframe system has a WebSphere MQ V7.0 for z/OS queue manager running. The warehouse is not interested in using Publish/Subscribe and they prefer to use point to point programming style in their applications.

The warehouse is wants to use call back to enable MQ to allocate the buffers with the appropriate length to receive the order messages. The order messages can vary in size from few hundreds to thousands of characters of text.

The warehouse component has two application programs:

- One application runs on the headquarter’s system, it retrieves copies of the orders that have been accumulated on the “MATT.RETAIL.WH.ORDERS” queue and it publishes the orders to topic “matt/warehouse/cat/<catalog id>”. Headquarters decided to use Pub/Sub instead of point to point communication with the warehouse because they have plans to establish several warehouses around the country as part of their business expansion plan. Pub/Sub enables new warehouses to subscribe and receive copies of the orders. Each warehouse decides if the order is for a store or a customer in their geographical area to process the order and deliver the goods. Warehouse systems require to receive copies of all orders for business contingency.
- The other application runs on the warehouse mainframe, it receives the orders from a local queue “MATT.WH.ORDERS”, process and display them. This program implements call back for asynchronous consume.

The following figure shows the programs and the interaction between the Headquarters and the Warehouse:

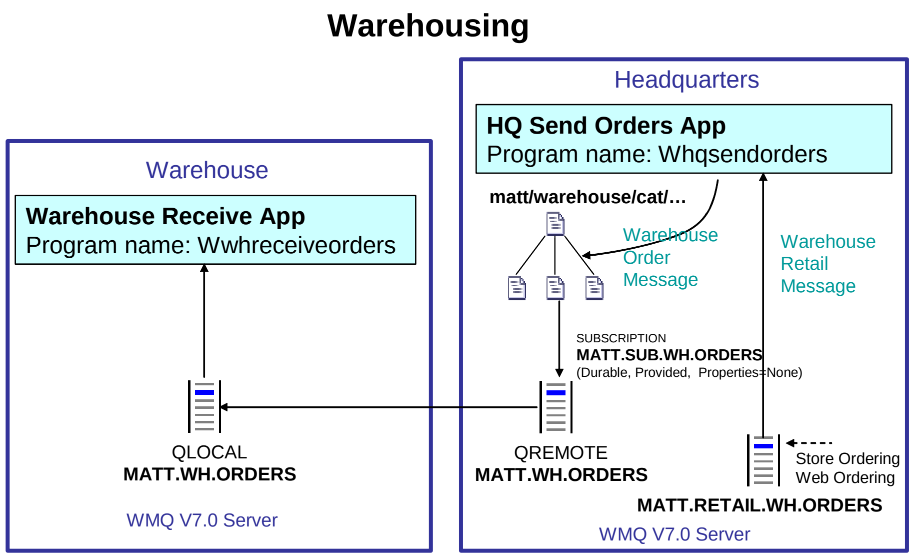


Figure 18-1 Warehousing scenario component

18.2 Deploying the Warehousing component

The Warehousing component is made of two application programs. One program runs on the headquarters system and the other runs on the warehouse mainframe.

The following are the programs that implement the warehouse component:

- HQ send orders (Whqsendorders): This is a C language program that gets messages accumulated in MATT.RETAIL.WH.ORDERS queue and publishes them to the topic matt/warehouse/<catalog id>.
- Warehouse receive orders (Wwhreceiveorders): This is a C language batch program that runs on the warehouse mainframe. This program get messages from local queue MATT.WH.ORDERS using a call back consumer function. Orders are displayed on the standard output of the program (SYSOUT). This program runs until it is cancelled by the operator or a message with the text ‘shutdown’ in the message body is received.

This component requires the creation of subscription MATT.SUB.WH.ORDERS using MQ explorer or MQSC commands. This subscription is made on the headquarters queue manager to topic matt/warehouse/#. This subscription is a non-managed durable subscription that has a subscriber queue defined as remote queue MATT.WH.ORDERS. This remote queue definition represents the queue MATT.WH.ORDERS on the z/OS queue manager QMWH running in the warehouse. This subscription is what converts publications made on the HQ queue manager into messages on a queue in the warehouse queue manager.

Sender and receiver channels are configured between queue managers QMHQ and QMWH.

18.2.1 Required MQ objects

The Warehousing component requires the following objects defined by the MQ administrator in the headquarters QMHQ queue manager:

- MATT.TOPIC.WAREHOUSE: This topic object defines the topic string matt/warehouse. This topic object is used by the HQ send orders program to publish orders to the warehouse (or warehouses in the future).
- MATT.SUB.WH.ORDERS: This is a non-managed durable subscription to topic object MATT.TOPIC.WAREHOUSE and to topic string #. The provided subscriber queue is MATT.WH.ORDERS that is a remote queue definition.
- MATT.WH.ORDERS: This is a queue remote object definition. The remote queue manager is QMWH and the remote queue is MATT.WH.ORDERS.
- MATT.RETAIL.WH.ORDERS: This is a local queue on the headquarters queue manager and that is defined for the Store Ordering component refer to Chapter 15, “Scenario: Store ordering with JMS” on page 283.

Example 18-1 Warehousing topics and queues for the headquarters queue manager

```
DEFINE QREMOTE(MATT.WH.ORDERS) +
   RNAME(MATT.WH.ORDERS) +
   RQMNAME(QMWH) +
   REPLACE

* Topics
DEFINE TOPIC(MATT.TOPIC.WAREHOUSE) +
   TOPICSTR('matt/warehouse') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE

* Subscriptions
DEFINE SUB(MATT.SUB.WH.ORDERS) +
   TOPICOBJ(MATT.TOPIC.WAREHOUSE) +
   TOPICSTR('#') +
   DESTCLAS(PROVIDED) +
   DEST(MATT.WH.ORDERS) +
   REPLACE
```

The following object is required on the warehouse QMWH queue manager:

- MATT.WH.ORDERS: This is local queue that receives the publications that match the subscription MATT.SUB.WH.ORDERS.

Transmission queues, sender and receiver channels are required to interconnect the QMHQ and QMWH queue managers.

Example 18-2 Warehousing queue for the z/OS queue manager

```
DEFINE QLOCAL(MATT.WH.ORDERS) +
   DEFPSIST(YES) +
   DEFSOPT(SHARED) +
   REPLACE
```

18.2.2 Installation of Warehousing component

The Warehousing programs are packaged in a file called CH18_Warehousing.zip. This file has source code, executable code and the nmake files to build the C language executables that runs in the headquarters server. The file also includes the source code and sample Job Control Language (JCL) to compile and link the z/OS programs.

This file has also a copy of the C language program receives orders (Wwhreceiveorders) in the warehouse that runs with a Windows queue manager.

Note to Reviewer: Please update final file names and the reference where to download the software.

The following are the instructions on how to install the executable code on Windows:

- Extract the contents of the compressed file into a temporary directory, for example C:\temp.
- The following directories are created:
  – C:\temp\Warehousing\windows\
  – C:\temp\Warehousing\zos\

- Change to directory C:\temp\Warehousing\windows\
- This windows directory contains the following files:
  - <program name>.c: This is the program the source code.
  - <program name>.exe: This is the executable code linked with server bindings library.
  - make.bat: This the batch command file that executes nmake to compile and link the programs. Execute this command if it necessary to recreate the executable code.
  - makewarehousing.txt : This is the nmake parameter file.

These programs are compiled with a C language compiler such as Microsoft Visual C++ Express Edition for Windows.

Note to Reviewer: Please add references of Windows, Microsoft Visual C++ compiler and make to the section of third party software.

The following are the instructions on how to install the program on z/OS:

- Change to directory C:\temp\Warehousing\zos\
- This directory has the following files:
  - WRCVORD.c: This is the source code of the receive orders program in C language.
  - CCOMPILE.jcl: This is a sample JCL to compile the C language program WRCVORD in z/OS.
  - CLINK.jcl: This is a sample JCL to link the C language program and create a load module (executable code).
  - WRCVORD.jcl: This is a sample JCL to run the WRCVORD program.
- In TSO, create libraries to store the JCL and the source code.
- Transfer the source code and JCL files to the z/OS TSO environment.
- In TSO, create libraries to store the object and load modules that are generated by the compilation and link of the program.
- Execute the CCOMPILE JCL to create the object module.
- Execute the CLINK JCL to link the object module and create the load module.
- Create the MQ objects required in the z/OS queue manager.
- Run the program using the WRCVORD JCL.

 - Check the output of the program for the messages that should the orders that have been received and processed.
 - This program runs a loop until a message with the word “shutdown” is received or the operator cancels it.

18.3 Running the Warehousing component

To run the Warehousing programs follow these steps:

- In TSO, execute the WRCVORD JCL to run the warehouse receive orders program.
- In Windows, open a command prompt.
- Change directory to the location of the executable code:

```
C:\>cd C:\temp\Warehousing\windows
```

- Execute program Whqsendorders:

```
C:\temp\Warehousing\windows>Whqsendorders.exe MATT.RETAIL.WH.ORDERS
QMHQ

**********************************
*  Whqsendorders program starts  *
**********************************
Connecting to Queue Manager 'QMHQ'
Opening warehouse retail queue 'MATT.RETAIL.WH.ORDERS' for input
```

- The program output shows the orders that are processed and published to the warehouse:

```
C:\temp\Warehousing\windows>Whqsendorders.exe MATT.RETAIL.WH.ORDERS
QMHQ
**********************************
*  Whqsendorders program starts  *
**********************************
Connecting to Queue Manager 'QMHQ'
Opening warehouse retail queue 'MATT.RETAIL.WH.ORDERS' for input
Got message
'matt/retail/cat/fresh/fruit/apples,5,ORDER0001,CUST0012'
Publishing message:
  Topic string 'matt/warehouse/cat/fresh/fruit/apples'
  Message data 'cat/fresh/fruit/apples,5,ORDER0001,CUST0012'
```

18.4 Verifying the Warehousing component

To verify the Warehousing component follow these steps:

- In z/OS TSO, execute JCL to run program WRCVORD.
- In a Windows command prompt, execute program Whqsendorders.exe
- In another Windows command prompt, execute the test command file testretailwhordersgen.bat to put some test orders on the MATT.RETAIL.WH.ORDERS queue:

```
C:\temp\Warehousing\windows>testretailwhordersgen.bat

C:\temp\Warehousing\windows>amqsput MATT.RETAIL.WH.ORDERS
0<testretailwhordersdata1.txt
Sample AMQSPUT0 start
target queue is MATT.RETAIL.WH.ORDERS
Sample AMQSPUT0 end

C:\temp\Warehousing\windows>pause
Press any key to continue . . .
```

- Check the output of the Whqsendorders.exe program:

```
C:\temp\Warehousing\windows>Whqsendorders.exe MATT.RETAIL.WH.ORDERS
QMHQ
**********************************
*  Whqsendorders program starts  *
**********************************
Connecting to Queue Manager 'QMHQ'
Opening warehouse retail queue 'MATT.RETAIL.WH.ORDERS' for input
Got message
'matt/retail/cat/fresh/fruit/apples,5,ORDER0001,CUST0012'
Publishing message:
  Topic string 'matt/warehouse/cat/fresh/fruit/apples'
  Message data 'cat/fresh/fruit/apples,5,ORDER0001,CUST0012'
Got message 'junk/cat/tinned/fish/salmon,3,ORDER0002,STOR0002'
Publishing message:
  Topic string 'matt/warehouse/cat/tinned/fish/salmon'
  Message data 'cat/tinned/fish/salmon,3,ORDER0002,STOR0002'
Got message 'cat/dry/flour/selfraising,2,ORDER0003,STOR0001'
Publishing message:
  Topic string 'matt/warehouse/cat/dry/flour/selfraising'
  Message data 'cat/dry/flour/selfraising,2,ORDER0003,STOR0001'
Got message 'cat/glass/spirit/vodka,6,ORDER0004,CUST0034'
Publishing message:
  Topic string 'matt/warehouse/cat/glass/spirit/vodka'
  Message data 'cat/glass/spirit/vodka,6,ORDER0004,CUST0034'
```

- In TSO, check the output of the WRCVORD program:

```
*************************************
*  Wwhreceiveorders program starts  *
*************************************
Main: Connecting to Queue Manager 'QMWH'
Main: Opening warehouse orders queue 'MATT.WH.ORDERS' for input
Main: Registering MQ call-back function
Main: Starting message consumer
Main: While loop sleeping every  10 seconds

WhOrdersConsumer: Message removed, Reason=0, DataLength=43,
  BufferLength=8192, GMO.ReturnedLength=43

<*> Warehouse has received an order:
  Product           'cat/fresh/fruit/apples'
  Quantity          '5'
  Order Id          'ORDER0001'
  Customer/Store Id 'CUST0012'

WhOrdersConsumer: Message removed, Reason=0, DataLength=43,
  BufferLength=8192, GMO.ReturnedLength=43

<*> Warehouse has received an order:
  Product           'cat/tinned/fish/salmon'
  Quantity          '3'
  Order Id          'ORDER0002'
  Customer/Store Id 'STOR0002'

WhOrdersConsumer: Message removed, Reason=0, DataLength=46,
  BufferLength=8192, GMO.ReturnedLength=46

<*> Warehouse has received an order:
  Product           'cat/dry/flour/selfraising'
  Quantity          '2'
  Order Id          'ORDER0003'
  Customer/Store Id 'STOR0001'

WhOrdersConsumer: Message removed, Reason=0, DataLength=43,
  BufferLength=8192, GMO.ReturnedLength=43

<*> Warehouse has received an order:
  Product           'cat/glass/spirit/vodka'
  Quantity          '6'
```

```
Order Id            'ORDER0004'
Customer/Store Id   'CUST0034'
```

18.5 Summary

The Warehousing component uses a durable non-managed that is created using administration commands or MQ Explorer to convert publications to a topic into point to point messages on a specific remote queue on a z/OS queue manager that is not running Pub/Sub.

Call back for asynchronous consume is demonstrated on a z/OS environment. The benefits of call back is that message buffers are allocated by the queue manager with the correct size for the message length.

A

Scenario preparation scripts

This appendix contains the following scripts, used for common scenario preparation:

- *QMHQobjects.txt*: the MQSC command script for headquarters queue manager objects definition.
- *QMWHobjects.txt*: the MQSC command script for warehouse queue manager objects definition.
- **`QMHQsetaut.bat`**: the Windows batch script for headquarters queue manager objects authorities settings, using setmqaut operational command.

For detailed information about these scripts refer to section 13.2, “WebSphere MQ objects setup” on page 259.

QMHQobjects.txt

This section contains the *QMHQobjects.txt* script.

Example: A-1 The script QMHQobjects.txt

```
* This script set up queue manager objects needed for scenario
* The script uses REPLACE parameter and can be run repeatedly

* Dead letter queue for Queue manager
ALTER QMGR DEADQ(SYSTEM.DEAD.LETTER.QUEUE)

* Listener
DEFINE LISTENER(QMHQ.TCP) +
   TRPTYPE(TCP) +
   CONTROL(QMGR) +
   PORT(1414) +
   REPLACE

* Queues
DEFINE QLOCAL(QMWH) +
   USAGE(XMITQ) +
   REPLACE
DEFINE QLOCAL(MATT.RETAIL.ORDERS) +
   REPLACE
DEFINE QLOCAL(MATT.RETAIL.RESPONSES) +
   REPLACE
DEFINE QLOCAL(MATT.RETAIL.WH.ORDERS) +
   PROPCTL(NONE) +
   REPLACE
DEFINE QLOCAL(MATT.SUPPLIER.QUOTES) +
        DEFPSIST(YES) +
   DEFSOPT(SHARED) +
   REPLACE
DEFINE QREMOTE(MATT.WH.ORDERS) +
   RNAME(MATT.WH.ORDERS) +
   RQMNAME(QMWH) +
   REPLACE

* Channels
DEFINE CHANNEL(QMHQ_TO_QMWH) +
   CHLTYPE(SDR) +
   TRPTYPE(TCP) +
   CONNAME('sam725wh(1420)') +
   XMITQ(QMWH) +
   REPLACE
DEFINE CHANNEL(QMWH_TO_QMHQ) +
   CHLTYPE(RCVR) +
   TRPTYPE(TCP) +
   REPLACE
DEFINE CHANNEL(QMHQ_SUPP_A) +
   CHLTYPE(SVRCONN) +
   TRPTYPE(TCP) +
   REPLACE
DEFINE CHANNEL(QMHQ_SUPP_B) +
   CHLTYPE(SVRCONN) +
   TRPTYPE(TCP) +
   REPLACE
DEFINE CHANNEL(QMHQ_STORES) +
   CHLTYPE(SVRCONN) +
   TRPTYPE(TCP) +
   REPLACE
DEFINE CHANNEL(QMHQ_NEWS) +
   CHLTYPE(SVRCONN) +
   TRPTYPE(TCP) +
   REPLACE

* Topics
DEFINE TOPIC(MATT.TOPIC.REQUESTQUOTE) +
   TOPICSTR('matt/requestquote') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE
DEFINE TOPIC(MATT.TOPIC.SUPPLIERQUOTE) +
   TOPICSTR('matt/supplierquote') +
   DEFPSIST(YES) +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE
DEFINE TOPIC(MATT.TOPIC.RETAIL) +
   TOPICSTR('matt/retail') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE
DEFINE TOPIC(MATT.TOPIC.WAREHOUSE) +
   TOPICSTR('matt/warehouse') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE
DEFINE TOPIC(MATT.TOPIC.NEWS.INTERNAL) +
   TOPICSTR('matt/news/internal') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE
DEFINE TOPIC(MATT.TOPIC.NEWS.PUBLIC) +
   TOPICSTR('matt/news/public') +
   PUB(ENABLED) +
   SUB(ENABLED) +
   REPLACE

* Subscriptions
DEFINE SUB(MATT.SUB.WH.ORDERS) +
   TOPICOBJ(MATT.TOPIC.WAREHOUSE) +
   TOPICSTR('#') +
   DESTCLAS(PROVIDED) +
   DEST(MATT.WH.ORDERS) +
   REPLACE

* End of script
```

QMWHobjects.txt

This section contains the *QMWHobjects.txt* script.

Example: A-2 The script QMWHobjects.txt

```
* This script set up queue manager objects needed for scenario
* The script uses REPLACE parameter and can be run repeatedly

* Dead letter queue for Queue manager
ALTER QMGR DEADQ(SYSTEM.DEAD.LETTER.QUEUE)

* Listener
DEFINE LISTENER(QMWH.TCP) +
   TRPTYPE(TCP) +
   CONTROL(QMGR) +
   PORT(1420) +
   REPLACE

* Queues
DEFINE QLOCAL(QMWH) +
   USAGE(XMITQ) +
   REPLACE
```

```
DEFINE QLOCAL(MATT.WH.ORDERS) +
   REPLACE

* Channels
DEFINE CHANNEL(QMWH_TO_QMHQ) +
   CHLTYPE(SDR) +
   TRPTYPE(TCP) +
   CONAME('sam725hq(1414)') +
   XMITQ(QMWH) +
   REPLACE
DEFINE CHANNEL(QMHQ_TO_QMWH) +
   CHLTYPE(RCVR) +
   TRPTYPE(TCP) +
   REPLACE

* End of script
```

QMHQsetaut.bat

This section contains the **`QMHQsetaut.bat`** script.

Example: A-3 The script QMHQsetaut.bat

```
@ECHO OFF
rem   This script set up queue manager authorities needed for scenario
rem   The script removes all authorities at first
rem   and can be run repeatedly

rem   Set up for group "hq"
setmqaut -m QMHQ -t qmgr -g hq +none
setmqaut -m QMHQ -t qmgr -g hq +connect
setmqaut -m QMHQ -t queue -n MATT.RETAIL.ORDERS -g hq -remove
setmqaut -m QMHQ -t queue -n MATT.RETAIL.ORDERS -g hq +get
setmqaut -m QMHQ -t queue -n MATT.RETAIL.RESPONSES -g hq -remove
setmqaut -m QMHQ -t queue -n MATT.RETAIL.RESPONSES -g hq +put
setmqaut -m QMHQ -t queue -n MATT.RETAIL.WH.ORDERS -g hq -remove
setmqaut -m QMHQ -t queue -n MATT.RETAIL.WH.ORDERS -g hq +get +put
setmqaut -m QMHQ -t queue -n MATT.WH.ORDERS -g hq -remove
setmqaut -m QMHQ -t queue -n MATT.WH.ORDERS -g hq +put
setmqaut -m QMHQ -t topic -n MATT.TOPIC.REQUESTQUOTE -g hq -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.REQUESTQUOTE -g hq +pub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.SUPPLIERQUOTE -g hq -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.SUPPLIERQUOTE -g hq +sub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.WAREHOUSE -g hq -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.WAREHOUSE -g hq +pub +sub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.RETAIL -g hq -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.RETAIL -g hq +pub +sub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.* -g hq -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.* -g hq +pub

rem   Set up for group "st"
setmqaut -m QMHQ -t qmgr -g st +none
setmqaut -m QMHQ -t qmgr -g st +connect
setmqaut -m QMHQ -t queue -n MATT.RETAIL.ORDERS -g st -remove
setmqaut -m QMHQ -t queue -n MATT.RETAIL.ORDERS -g st +put
setmqaut -m QMHQ -t queue -n MATT.RETAIL.RESPONSES -g st -remove
setmqaut -m QMHQ -t queue -n MATT.RETAIL.RESPONSES -g st +get
setmqaut -m QMHQ -t topic -n MATT.TOPIC.RETAIL -g st -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.RETAIL -g st +sub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.INTERNAL -g st -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.INTERNAL -g st +sub

rem   Set up for group "sp"
setmqaut -m QMHQ -t qmgr -g sp +none
setmqaut -m QMHQ -t qmgr -g sp +connect
setmqaut -m QMHQ -t topic -n MATT.TOPIC.REQUESTQUOTE -g sp -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.REQUESTQUOTE -g sp +sub
setmqaut -m QMHQ -t topic -n MATT.TOPIC.SUPPLIERQUOTE -g sp -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.SUPPLIERQUOTE -g sp +pub

rem   Set up for group "matt"
setmqaut -m QMHQ -t qmgr -g matt +none
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.PUBLIC -g matt -remove
setmqaut -m QMHQ -t topic -n MATT.TOPIC.NEWS.PUBLIC -g matt +sub

rem   End of script
```

B

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

ftp://www.redbooks.ibm.com/redbooks/SG247583

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247583.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*

All.zip	All zipped Code Samples
CH13_ ScenarioPrep.zip.zip	Chapter 13 zipped Code Samples
CH14_SupplierPricing.zip	Chapter 14 zipped Code Samples
CH15_StoreOrdering.zip	Chapter 15 zipped Code Samples
CH16_ News.zip	Chapter 16 zipped Code Samples
CH18_Warehousing.zip	Chapter 18 zipped Code Samples

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Glossary

Application server - A managed environment within which applications are deployed and run, with access to a defined set of functionality which may include a messaging facilities, such as WebSphere MQ.

Broker - In a publish/subscribe messaging model, a broker maintains information about topics and the subscribers upon those topics. When a publisher publishes information on a topic to the broker, the broker distributes that information to all registered subscribers.

Business critical data - Data which is not stored elsewhere in the system. If this data is lost then important information, or a change in state within the system, is lost.

Certificate Authority (CA) - A entity, identified by their public certificate, which is trusted to sign the certificates of others.

Certificate repository - A password protected store containing the public certificates of trusted certificate authority (CA), and also personal certificates, including their private keys.

Certificate Revocation List (CRL) - A list of revoked certificates, previously signed by a certificate authority, which have been compromised and should not be trusted. Usually queried from a Lightweight Directory Access Protocol (LDAP) server.

Channel - A network communications link between two queue managers over which messages flow, or a network communications link between an application and a queue manager over which Message Queuing Interface (MQI) commands flow.

CipherSpec - A method for specifying a CipherSuite which assume RSA is used as teh key exchange protocol.

CipherSuite - A method for specifying the key exchange, encryption and message authentication code (MAC) algorithms to use for securing a channel with the Secure Sockets Layer (SSL) or Transport Layer Security (TSL).

Client application - An application connecting to a WebSphere MQ queue manager over a network.

Cluster - See queue manager cluster.

Cluster message channel - A message channel between two queue managers within the same queue manager cluster.

Data conversion - The process of converting the binary representation of characters and numbers from that of one environment to that of another.

Disaster recovery - The process of recovering data, and restoring access services, after a significant event which impacts multiple nodes in a system

Distributed message channel - A message channel between two queue managers, where all messages are transferred from a single transmission queue on one queue manager, to destination queues on the remote queue manager.

Event information - Information that describes an event which has occurred, and may require action.

Exactly once delivery - An assurance provided by a messaging infrastructure that a message arrives at its destination, that it arrives once, and that it arrives once only.

Failover - The process of making the data held by a node, and the services provided by that node, available on a different node.

Global unit of work - A unit of work which includes actions upon multiple different resources, which may include WebSphere MQ and database products. This unit of work is coordinated by a transaction manager.

High availability - High availability encompasses the concepts of service availability, and message availability, in a message queuing environment.

High availability cluster - A mechanism to automatically failover the data held by a node, and the services provided by that node, in the event that that node experiences a planned or unplanned outage.

Hub and spoke architecture - A WebSphere MQ infrastructure architecture in which services are provided by a small number of hub queue managers, and access to those services is extended through a larger number of intermediate spoke queue managers interconnected with those hub queue managers.

IBM Message Service Client (XMS) - An application programming interface for the C and C++ programming languages, and the .NET environment, which is consistent with the Java Message Service application programming interface.

Java Message Service (JMS) - An industry standardised application programming interface for the Java programming language, which is part of the Java 2 platform Enterprise Edition standard.

Load - The number of attempts being made to request a service within a given time interval.

Message - A piece of information, with addressing or other meta information associated, that can be passed between software components.

Message availability - If a failure occurs on a node through which messages flow, whether those messages be recovered if that node fails, and how quickly they become available

Message channel - A network communications link between two queue managers over which messages flow.

Message Channel Agent (MCA) - A component of a WebSphere MQ queue manager, or a WebSphere MQ client product, which forms one half of a channel, establishing network communications with, or responding to network communications from, a partner MCA.

Message descriptor - See WebSphere MQ Message Descriptor (MQMD)

Message queuing - A middleware technique which allows unlike software components to interact asynchronously through a queue.

Message queuing interface (MQI) - The core application interface used to interact with a WebSphere MQ infrastructure.

Message queuing interface (MQI) channel - A network communications link between an application and a queue manager over which Message Queuing Interface (MQI) commands flow.

Middleware - A software infrastructure layer between applications and the infrastructure components they interact with; which is common to multiple nodes in a system, and simplifies interaction between the unlike software and hardware components which reside on those nodes.

Object Authority Manager (OAM) - A component of a WebSphere MQ queue manager that performs authority checking.

Outage - A period of time when a service, or services, provided by a system are unavailable.

Performance - The time taken between submitting a request for a service and completion of that service. How the start and end points of a service are determined are specific to the function being performed by the service.

Personal certificate - A public certificate which can be used to identify an entity, combined with the private key for that certificate.

Planned outage - Periods when a service, or services, are unavailable in order to perform planned work upon those services.

Point to point messaging - The sending of messages from one location to a single destination that is determined based upon addressing information provided by the sender of the message.

Polling - Repeatedly requesting a piece of information at regular intervals, in order to detect changes in that information

Production environment - An environment through which real services are made available within and/or outside of the business.

Proxy - An interface between an existing service, usually with a proprietary interface, and a middleware layer that is used by other nodes in the system to access that service.

Publish/subscribe messaging - A model of messaging in which the producers of information do not have direct knowledge of the consumers of that information, which may be zero or many.

Publisher - In a publish/subscribe messaging model, a publisher produces information on a particular topic which is distributed to registered subscribers on that topic by a broker.

Quality assurance environment: An environment created to simulate a production environment, for testing and development of application and infrastructure changes.

Query data - Transient data being sent through a system, derived from data that is stored safely within the system.

Queue - A container for messages, from which messages are usually retrieved in first-in-first-out order, which can be used as an asynchronous buffer between two software components.

Queue manager - Queue managers are the interconnected nodes within a WebSphere MQ infrastructure that maintain the messages and queues, provide data integrity, and provide applications with access to the infrastructure to send and receive messages.

Queue manager cluster - A mechanism provided by WebSphere MQ to interconnect queue managers in a flexible way, which simplifies administration and provides workload balancing facilities for scalability and service availability.

Queue name resolution - The action performed by a queue manager whenever an application or channel attempts to open a queue, in order to put a message upon a queue hosted by that queue manager, or to send a message through that queue manager.

Queue sharing group - A feature of WebSphere MQ for z/OS which allows applications connected to multiple queue managers, running on different z/OS systems within a sysplex, to get and put messages to the same queue.

Request/reply messaging - Asynchronous communication between two software components, in which a request message is sent and a reply message is returned following processing of the request.

Resource manager - A component which, under the control of a transaction manager, manages an individual resource which is participating in a global unit of work

Scalability - How easily the capacity of the system can be increased to cope with increased load, and how this affects performanceSecure Sockets Layer (SSL) - An industry standardised technology to provide authentication and secure communication.

Secure Sockets Layer (SSL) - An industry standardised technology to provide authentication and secure communication.

Security of access - Ensuring that services are only accessible by those entities authorised to do so.

Security of communications - Ensuring that sensitive information cannot be intercepted or tampered with during communication

Send and forget messaging - The sending of messages without requiring a reply upon processing of those messages; hence relying on the exactly once delivery assurance of the message queuing infrastructure to deliver the message.

Service availability - If a planned or unplanned outage affects nodes in the system which provide a service, whether the system as a whole can continue to provide that service

State information - Information that changes over time, but only has one value at any point in time.

Statement of environment (SOE) - Details of the supported versions of operating systems, compilers and other software components which interact with the WebSphere MQ product. SOEs are available through the WebSphere MQ support Web page.

Subscriber - In a publish/subscribe messaging model, a subscriber registers with a broker to receive all information published on a particular topic.

SupportPac - A package of additional functionality or documentation for the WebSphere MQ product, distributed through the IBM SupportPacs Web page.

Topic - In a publish/subscribe messaging model, a topic is used group information so that publishers which produce information on a topic can be loosely coupled with subscribers that consume information on that topic.

Transaction - The mechanism by which multiple actions, possibly upon multiple resources, can be grouped together in a unit of work.

Transaction manager - The component which manages the resources participating in a global unit of work.

Transport Layer Security (TLS) - An industry standardised technology to provide authentication and secure communication.

Unit of work - A logical grouping of actions, which must either all succeed or all fail.

Unplanned outage - Periods when a service, or services, become unavailable unexpectedly.

Web services - A standardised way to describe and invoke services.

WebSphere MQ Message Descriptor (MQMD) - A data structure, associated with each WebSphere MQ message, that contains meta information associated with that message; such as identifying and type information.

WebSphere MQ object model - A defined set of classes, methods and properties to interact with WebSphere MQ which are implemented for multiple object oriented programming languages, including Java and C++, building upon the facilities provided by the Message Queuing Interface (MQI).

Abbreviations and acronyms

abbreviation1	Description1
abbreviation2	Description2
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
abbreviation3	Description3
abbreviation4	Description4

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 352. Note that some of the documents referenced here may be available in softcopy only.

- *????full title???????,* xxxx-xxxx
- *????full title???????*, SG24-xxxx
- *????full title???????*, REDP-xxxx
- *????full title???????*, TIPS-xxxx

Other publications

These publications are also relevant as further information sources:

- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx
- *????full title???????,* xxxx-xxxx

Online resources

These Web sites are also relevant as further information sources:

- Description1

  `http://????????.???.???/`
- Description2

  `http://????????.???.???/`
- Description3

  `http://????????.???.???/`

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

Redbooks

WebSphere MQ V7.0 Features and Enhancements

(1.5" spine)
1.5"<-> 1.998"
789 <->1051 pages

Redbooks

WebSphere MQ V7.0 Features and Enhancements

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

Redbooks

WebSphere MQ V7.0 Features and Enhancements

(0.5" spine)
0.475"<->0.875"
250 <-> 459 pages

Redbooks

WebSphere MQ V7.0 Features and Enhancements

(0.2"spine)
0.17"<->0.473"
90<->249 pages

(0.1"spine)
0.1"<->0.169"
53<->89 pages

Redbooks

WebSphere MQ V7.0 Features and Enhancements

(2.5" spine)
2.5"<->nnn.n"
1315<-> nnnn pages

Redbooks

WebSphere MQ V7.0 Features and Enhancements

(2.0" spine)
2.0" <-> 2.498"
1052 <-> 1314 pages

WebSphere MQ V7.0 Features and Enhancements

Integrated Publish/Subscribe engine and new MQI functions

Improved JMS MQ integration and MQ Client enhancements

Scenario with sample code

Introducing new WebSphere MQ V7.0 features:

- Both MQI and JMS APIs
- RAS features within JMS

Exploring the new features:

- Publish/Subscribe
  Consolidating Pub/Sub domain
  Distributed pub/sub
  Available on z/OS
- MQI enhancements
  Message selectors
  Message properties
  Call-back allows asynchronous consumption
- Client enhancements
  Asynchronous put
  Full duplex
  Conversation sharing
  Read ahead
  Channel instance limits
- Interaction between JMS and MQI Applications
- MQ Explorer enhancements including JMS administration
- MQ HTTP bridge
- z/OS enhancements
- Migration considerations

SG24-7583-00 ISBN