WebSphere. software

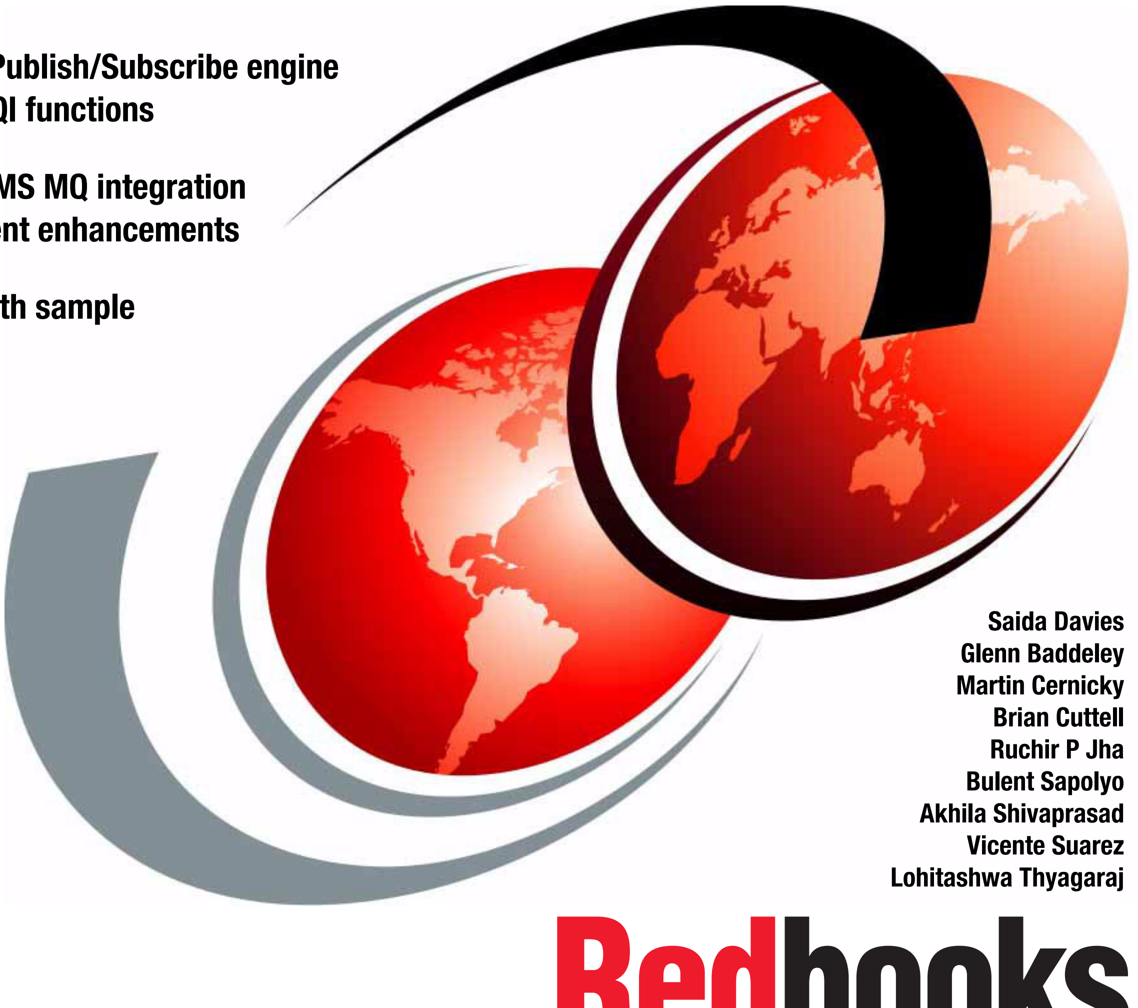
IBM

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
Akhila Shivaprasad
Vicente Suarez
Lohitashwa Thyagaraj

Redbooks

ibm.com/redbooks

International Technical Support Organization

WebSphere MQ V7.0 Features and Enhancements

January 2009

SG24-7583-00

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

First Edition (January 2009)

This edition applies to:

Version	Release	Modification	Product name
7	0	0	WebSphere MQ
6	0	2.2	WebSphere MQ Client (SupportPac MQC6)
6	1	0	WebSphere Application Server
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

Contents

Figures

Tables

Examples

Notices

Trademarks

Preface

This IBM® Redbooks® publication is divided into three parts:

- Part 1, "Introduction" on page 1, provides an introduction to message-oriented middleware and the WebSphere® MQ product. We discuss the concept of messaging, 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, "WebSphere MQ V7.0 enhancements and changes" on page 41, 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 listed below.

  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
    - Callback 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, "Scenario" on page 253, contains a scenario that 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 379.

The information included in this book 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.

Figure 1 The team (from left): Akhila, 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 publications and Redpapers publications 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 the z/OS operating system, and a detailed working knowledge of both IBM and independent software vendor 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 to and participates in their meetings.

Glenn Baddeley is a Specialist in WebSphere MQ and has been with IBM Global Technology Services Australia for 10 years. He leads the support of WebSphere MQ on several large outsourcing contracts in the Asia/Pacific region. Glenn performs architecture design and consulting, and strategic planning. He also sets standards and writes specialized documentation. He does 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 14 years as a systems programmer, designing, developing, and supporting customized middleware solutions and ISV Operating System security extensions. Glenn has a bachelor’s 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 IBM Software Services in the Czech Republic. He has 17 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 25 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 in the early 1990s and was one of the original members of the MQ development team on MVS. Since then, he has worked on various assignments including IBM UK's graduate program. Brian has more recently specialized 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 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 that 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. He creates strategies that 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 25 years of 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 company as an architect, consultant, and technical leader with IBM Asia Pacific for DBDC products on z/OS and WebSphere MQ, and 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 within IBM Global Technology Services, his role includes setting software direction and migration paths for outsourced customers and implementing WebSphere MQ. Bulent also designed the architecture and scope of 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.

Akhila Shivaprasad works as an Information Management Software Engineer for IBM India Software Labs, Bangalore. She joined IBM over three years ago. She is part of the WebSphere MQ PreGA and PostGA teams and is responsible for testing WebSphere MQ fix packs (MDVs) on distributed platforms, especially AS400. She has worked for over 30 LifeCyleRequests (LCRs) and has six PRBs to her credit. Akhila holds a degree in Electronics and Communication Engineering from Vishveshwaraiah Technological University. As a University Relationship Manager and Subject Matter Expert, she works alongside IBM University Relationship and IBM Academia. She was the originator of the IBM Messaging Blog and ensures its continuity. Akhila supports IBM India Software Labs-Shakti for various Women in Technology activities.

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 Redbooks 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. In addition, Vincente has developed and published support packs for WebSphere Message Broker and articles in IBM developerWorks®. 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 Technical Lead for the WebSphere Application Server and WebSphere MQ Java/Java™ Message Service Level-3 service team, where 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 and Distribution, Software Sales, IBM Hursley, for his support in facilitating the residency in Hursley.

Bruce Coughtrie
Manager SWG BetaWorks, IM, WebSphere and Rational®, IBM Sales and Distribution, Software Sales, IBM Hursley, for his support in facilitating the residency in Hursley.

Tasnim Kapasi
Gap year student completing her Industry experience with IBM. Her contribution was to set up specific variables and build the glossary and the index incorporated in this 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. Tasnim has achieved excellent A-levels in English literature, economics, and art. She is currently studying IT and English at London University.

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

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

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

Keith Watson
WebSphere MQ Base and Clients Development Manager, Software Group, Application and Integration Middleware Software, IBM Hursley, UK

Gavin Beardall
WebSphere MQ Software Developer, Software Group, Application and Integration Middleware Software, IBM Hursley UK

Matthew Whitehead
Senior Inventor, WebSphere MQ Bridge for HTTP, Software Group, Application and Integration Middleware Software, IBM Hursley UK

Luke Saker
IBM Software Group, WebSphere MQ JMS development, Java Software Developer, Application and Integration Middleware Software, IBM Hursley, UK

Adrian Dick
Software Developer, Software Group, WebSphere MQ and Enterprise Service Bus Development, Application and Integration Middleware Software, IBM Hursley UK

Barry Spiers
WebSphere MQ Base Distributed Development and Performance Manager, Software Group, Application and Integration Middleware Software, IBM Hursley UK

Jonathan Rumsey
Lead System i® Developer WebSphere MQ Software Developer, Software Group, Application and Integration Middleware Software, IBM Hursley UK

Become a published author

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 begins 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).

This part consists of:

1

Overview

This chapter provides an overview of this entire IBM Redbooks publication. It covers the scope of material, the intended audience, and assumptions made concerning the reader. This chapter contains the following sections:

- 1.1, “Executive summary” on page 4
- 1.2, “The scope of this book” on page 4
- 1.3, “Intended audience” on page 4
- 1.4, “What is covered in this book” on page 5
- 1.5, “What is not covered in this book” on page 6
- 1.6, “Assumptions” on page 6

1.1 Executive summary

The power of WebSphere MQ is its flexibility combined with reliability, scalability, and security. This flexibility provides a large number of design and implementation choices. Making informed decisions from this range can simplify the development of applications and the administration of a WebSphere MQ messaging 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 substantial array 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 enhancements made in WebSphere MQ V7.0 and the concepts that must be understood when developing applications that use the new features.

A broad understanding of the product features is key to making informed design and implementation choices for both the infrastructure and the applications that access it.

Details of new areas of function for WebSphere MQ V7.0 are introduced throughout this book, such as the WebSphere MQ Explorer, Publish/Subscribe integration, MQ Java Message Service providers, Message Queue Interface extensions, administration enhancements, MQ Client, and changes for the z/OS platform. Some installation and migration considerations when moving to WebSphere MQ V7.0 from prior releases are also discussed.

1.3 Intended audience

This book provides details about IBM WebSphere MQ V7.0 product features and enhancements required for individuals and organizations 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

The three distinct parts of this book provide an sample environment in which to gain an understanding of WebSphere MQ concepts, with a focus on the Publish/Subscribe integration with Java Message Service and the Message Queue Interface provided in Version 7.0 of the product. The three parts of this book are:

- Part 1, “Introduction” on page 1:
  - Chapter 1, “Overview” on page 3
  - Chapter 2, “Concepts of messaging” on page 7
  - Chapter 3, “Introduction to WebSphere MQ” on page 19

  These chapters provide an overview of this book and introduce the concepts of messaging. The basic features of WebSphere MQ are presented and the new features of WebSphere MQ V7.0 are summarized and positioned in contemporary IT architectures.
- Part 2, “WebSphere MQ V7.0 enhancements and changes” on page 41:
  - Chapter 4, “Publish/Subscribe integration” on page 43
  - Chapter 5, “WebSphere MQ Client enhancements” on page 59
  - Chapter 6, “Message Queue Interface extensions” on page 85
  - Chapter 7, “WebSphere MQ Java Message Service enhancements” on page 137
  - Chapter 8, “Administration enhancements” on page 153
  - Chapter 9, “Publish/Subscribe management” on page 197
  - Chapter 10, “WebSphere MQ Bridge for HTTP” on page 225
  - Chapter 11, “z/OS enhancements” on page 235
  - Chapter 12, “Installation and migration” on page 243

  These chapters discuss the new features and enhancements in WebSphere MQ V7.0, which provide the full native function support for Publish/Subscribe. administration, the MQ client, and relevant changes to other components of WebSphere MQ are also covered.
- Part 3, “Scenario” on page 253:
  - Chapter 13, “Scenario overview” on page 255
  - Chapter 14, “Scenario preparation” on page 269
  - Chapter 15, “Scenario: Supplier pricing using Pub/Sub” on page 287
  - Chapter 16, “Scenario: Store ordering with JMS” on page 303

- Chapter 17, “Scenario: News using client” on page 325
- Chapter 18, “Scenario: Web ordering over HTTP” on page 343
- Chapter 19, “Scenario: Warehousing using call back” on page 361

These chapters present a scenario that illustrates how to use the new features described in Part 2, “WebSphere MQ V7.0 enhancements and changes” on page 41.

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 the role of WebSphere MQ architects, programmers, and administrators.

1.6 Assumptions

Part 1, “Introduction” on page 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.

Part 2, “WebSphere MQ V7.0 enhancements and changes” on page 41, and Part 3, “Scenario” on page 253, concentrate on all the new features and enhancements in WebSphere MQ V7.0 in technical detail and illustrate most of them in a multi-faceted scenario. This assumes a good knowledge of many of the basic features that were introduced in previous versions of WebSphere MQ. Refer to the Redbooks publication *WebSphere MQ V6 WebSphere MQ V6 Fundamentals*, SG24-7128, for a comprehensive introduction to the foundation features of WebSphere MQ. This is available on the IBM Web site at:

`http://www.redbooks.ibm.com/abstracts/sg247128.html`

2 Concepts of messaging

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

It contains the following sections:

2.1 Enterprise messaging

Enterprise messaging is a general term to describe the orchestrated exchange of business data between disparate applications in complex environments. The main features of enterprise messaging are that it provides application decoupling, hides operating system specifics, and relieves applications from dealing with communications protocols. The common building block of exchanging data is the message, which can be records, files, events, Web services, requests, and responses, to name a few possibilities.

Enterprise messaging is emerging as a powerful, robust, high-capacity integration methodology, alongside established techniques such as Remote Procedure Call (RPC), Common Object Request Broker Architecture (CORBA), and other distributed application and client-server architectures. Many of these use synchronous operations, meaning that the receiving application must be available at the time that the requesting application passes the data. Enterprise messaging provides an environment to send and receive message asynchronously.

A means to enable asynchronous messaging is to provide a queuing facility that will *store and forward* messages from a sending application to be delivered to a receiving application that may not be in operation at the same time.

Message-oriented middleware (MOM)

A software product that provides enterprise messaging services is usually called message-oriented middleware.

There are many MOM software products available. WebSphere MQ is the IBM implementation. Chapter 3, "Introduction to WebSphere MQ" on page 19, provides an introduction to WebSphere MQ.

As the number of applications participating in the messaging infrastructure increases, the need for standardized methods of exchanging messages is necessary so that two or more applications can cooperate in their exchange of messages. This is where Java Message Service (JMS) is often utilized. An introduction to JMS is provided 2.3, "Java Message Service" on page 13.

2.2 Introducing Publish/Subscribe

Publish/Subscribe (Pub/Sub) is a messaging paradigm to send and receive messages asynchronously without the applications needing to know who or where their partner applications are or how they process data.

2.2.1 Publish/Subscribe

Pub/Sub is primarily intended for situations where a single message may need to be distributed to multiple applications. The main advantage over other message distribution methods is that it keeps the publisher separated from the subscribers. This means that the publisher application does not need to have any knowledge of either the subscriber application's existence or how they use the published information. Likewise, the subscriber applications do not need to know anything about the publisher applications. They have no dependencies on each other. This decoupling of publishers and subscribers allows for greater scalability and a dynamic processing 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 Pub/Sub in their application programming interface (API). For example, Java Message Service supports both the Publish/Subscribe and the message queue models. An example of a distributed execution model that can be applied to Pub/Sub design is data replication for sharing of information among various applications running on multiple machines.

The following example describes how the Publish/Subscribe model can be utilized.

A sports Web site that is dedicated to providing information about various games (cricket, football, hockey, rugby, and so on) provides hourly and daily updates on the events happening at sports fields across the world. There may be many users registered with this Web site who are interested in receiving regular updates on the sports of their choice. Some users can choose to subscribe and receive information for only a specific topic such as cricket, while another user may be interested in both cricket and football, and another user may be interested in all the topics.

The Web site providing the hourly or daily updates of the sports event is the publisher and the users subscribed to this Web site and consuming the messages are the subscribers. If there are multiple subscribers registered, it would be very difficult for the publisher to manage the sending of 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 to receive them. 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. In this case, the publisher must publish only one message to the topic for every update irrespective of how many subscribers are subscribed.

When the subscriber logs into the Web site, all the messages subsequently published to the topic for which the subscriber has subscribed are received instantly. The same subscriber can subscribe to more than 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 implementation, publishers publish messages to topics and there are one or more subscribers subscribed to topics to receive the messages. Subscribers may need to process a subset of the total messages published. Subscribers can filter messages in two ways, either by topic-based or by content-based selection.

Topic based

In this model the publishers publish messages to various topics and subscribers subscribe to the particular topics of their interest to receive messages. All the subscribers to each topic receive identical copies of the published messages on that topic.

For example, Figure 2-1 shows two publishers publishing the message `Welcome to world of colors` to the topic Color and all the subscribers that are subscribed to the topic receive both published messages. In this case, the subscribers do not discriminate between the publishers.

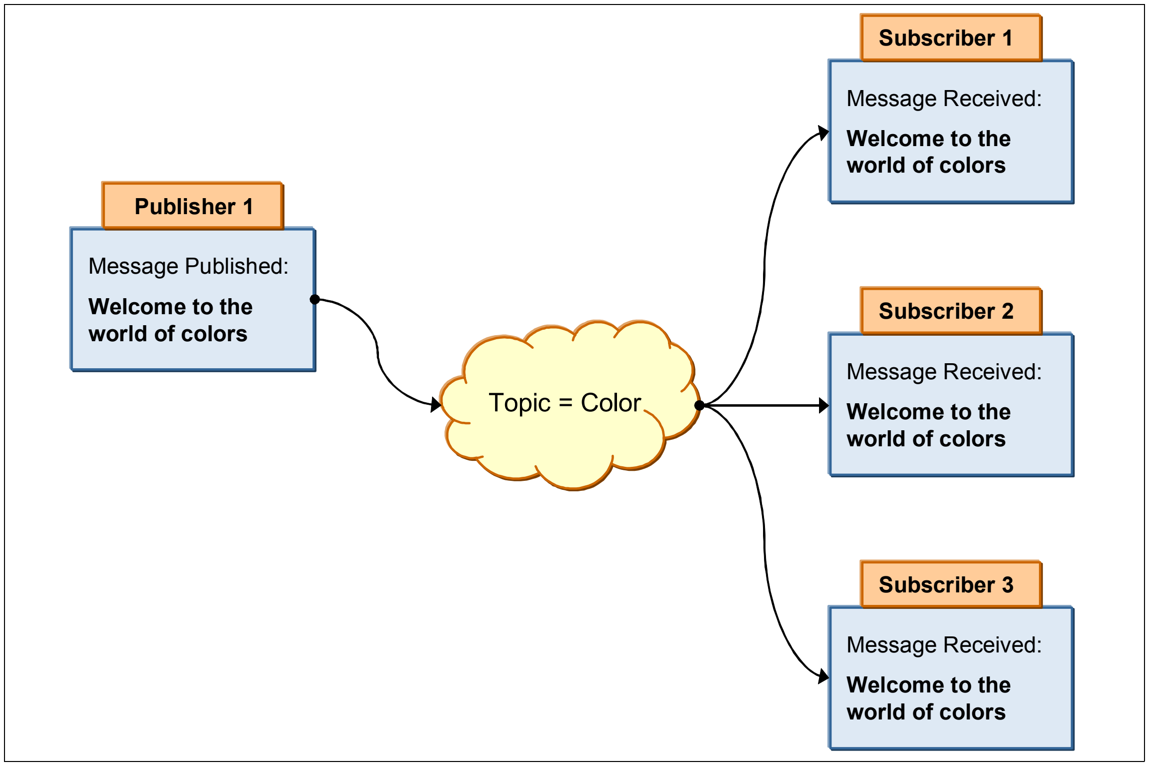


Figure 2-1 Topic-based selection

Content based

In this model the publishers publish messages to a particular topic and the subscribers that are subscribed to this topic filter the messages that they want based on the contents of the message. The subscriber can specify constraints for the kind of messages that they want to receive.

For example, in Figure 2-2, the first publisher publishes a message called `Welcome to the world of colors` and also sets a 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 that match their constraints. If no constraints are specified then that subscriber receive all messages published to that topic.

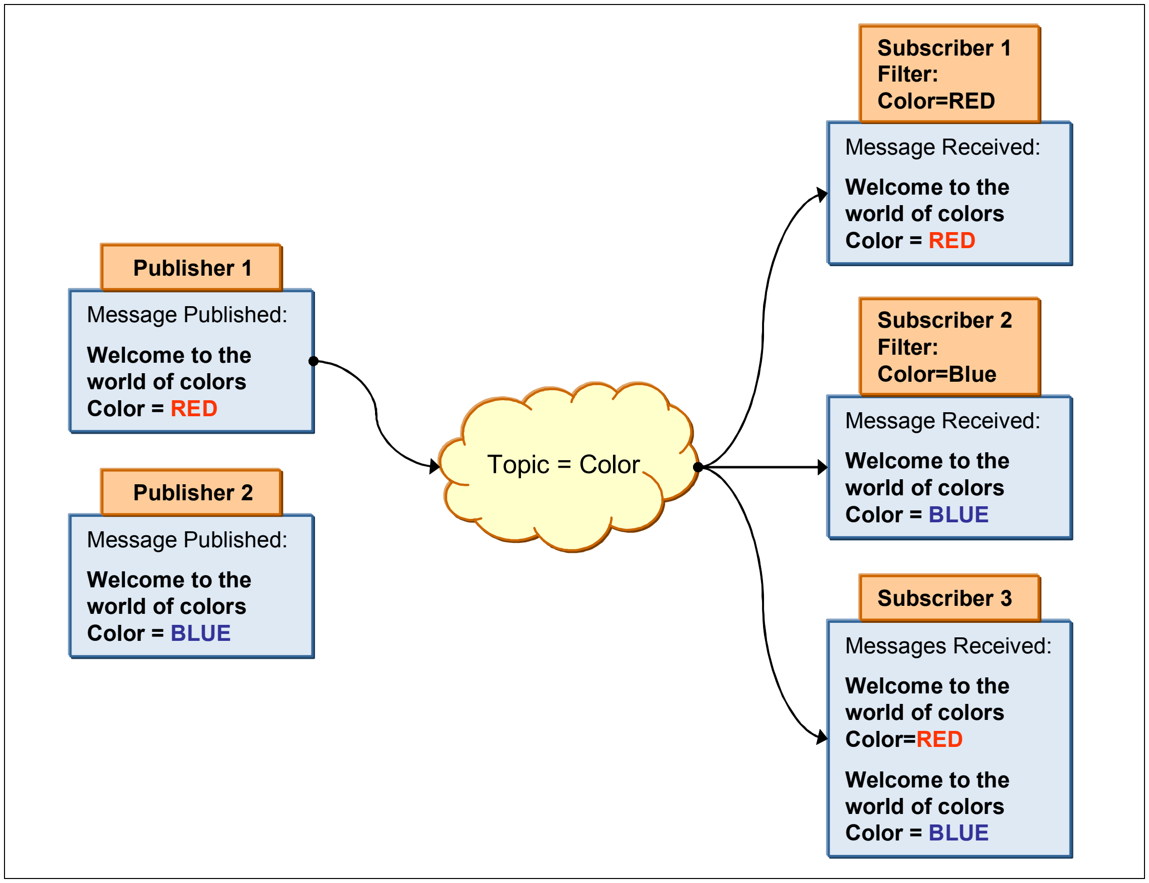

Publisher 1
Message Published:
Welcome to the world of colors Color = RED
Publisher 2
Message Published:
Welcome to the world of colors Color = BLUE
Topic = Color
Subscriber 1 Filter: Color=RED
Message Received:
Welcome to the world of colors Color = RED
Subscriber 2 Filter: Color=Blue
Message Received:
Welcome to the world of colors Color = BLUE
Subscriber 3
Messages Received:
Welcome to the world of colors Color=RED
Welcome to the world of colors Color = BLUE

Figure 2-2 Content-based selection

Some systems support a hybrid of the two. Publishers publish messages to topics while subscribers register content-based subscriptions to one or more topics.

2.2.3 Advantages

In this section we discuss the advantages.

Loosely coupled

Pub/Sub applications are loosely coupled, that is, the publisher and subscriber applications need not know the existence of each other. With the topic being the only common focus, publisher and subscriber applications operate independently and messages are passed asynchronously across the messaging system.

Compared with tightly coupled client-server architectures, there is less dependence between applications. They do not need to be running at the same time or in a specific location.

Pub/Sub also overcomes the single destination limitation of point-to-point messaging by providing a dynamic architecture where many applications can subscribe to receive the same messages.

Scalable

Pub/Sub provides the opportunity for improved scalability over other asynchronous messaging methodologies through parallel operation of producers and consumers of messages, topic-based routing, control over durability of subscriptions, and control over retention of published messages.

2.3 Java Message Service

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

2.3.1 Java Messaging

Java Message Service is a set of interfaces and associated semantics that define how a JMS client application accesses the facilities of an underlying enterprise messaging product. Messaging is recognized as an essential tool for building enterprise applications and e-commerce systems, and JMS provides a common methodology for Java programs to create, send, receive, and read enterprise messages.

Rather than allowing the applications to communicate directly with each other, applications send messages to a message server, which in turn delivers the messages to recipient applications. This might seem like an extra, unnecessary layer of software, but the advantages of using a message service provider 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 friends 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 communication issues. It also greatly enhances scalability.

JMS defines several interfaces for message services but no particular implementation. This gives great flexibility for vendors to implement message services the way that they want but still allows programmers to develop JMS applications that are largely independent from specific JMS messaging providers. This way the programmers do not have to rewrite their entire application when changing the underlying messaging system.

JMS offers two different messaging paradigms:

- Point-To-Point
- Publish/Subscribe

2.3.2 Point-To-Point model

The Point-To-Point model is built around the concept of message queues that have the capability of storing and forwarding messages for communication between coupled applications. An application sends messages to a queue and a partner application then receives the messages. Figure 2-3 depicts how a simple Point-To-Point model works.

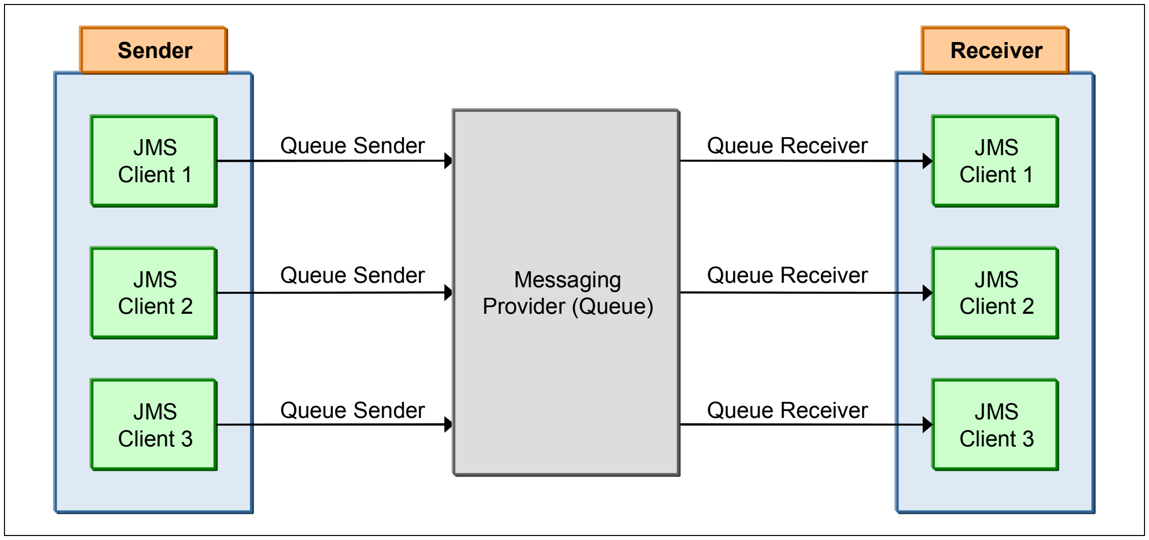


Figure 2-3 Point-To-Point model

2.3.3 Publish/Subscribe model

The Pub/Sub model is built around the concept of topics. A publisher application sends messages to a topic. A subscriber application receives messages from one or more topics. Messages are not generally stored on a topic. They are delivered to one or more subscriber applications if they are running. See 2.2, “Introducing Publish/Subscribe” on page 8, for more information.

2.3.4 Advantages of JMS

The advantages of JMS are:

- JMS allows programmers to write messaging applications that are independent of the underlying message provider implementation.
- Message provider independence allows the IT infrastructure to be upgraded, reorganized, and scaled without requiring changes to the JMS programs.
- Since JMS is purely a Java component, the platform in which the application runs is irrelevant, and it may operate and be moved between multiple platforms in the enterprise.

2.4 Position messaging in service-oriented architecture

Service-oriented architecture 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, as shown in Figure 2-4. SOA is an architectural style that makes this possible. For more information about SOA refer to the following link:

http://www.ibm.com/soa

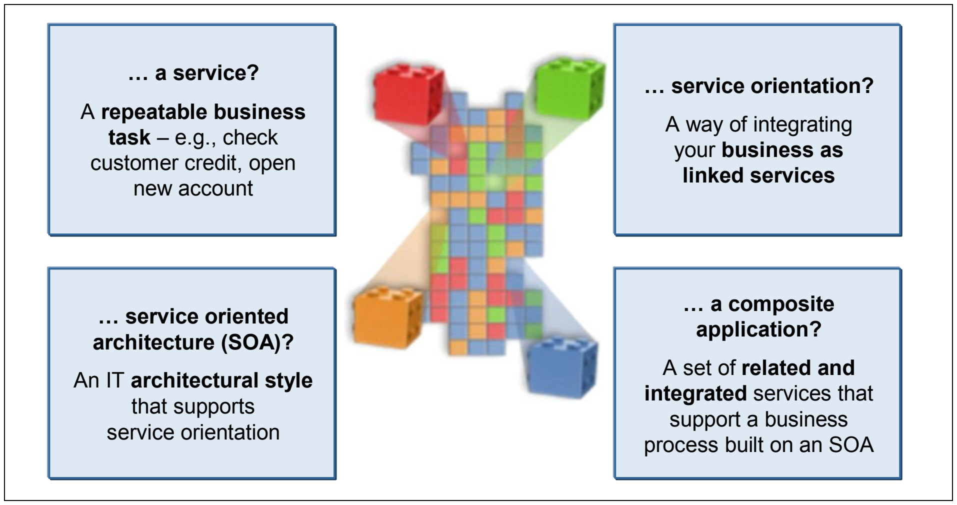

… a service?
A repeatable business task – e.g., check customer credit, open new account
… service orientation?
A way of integrating your business as linked services
… service oriented architecture (SOA)?
An IT architectural style that supports service orientation
… a composite application?
A set of related and 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 technology 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 and customer engagements and allowed the customer the flexibility to choose any entry point based on their necessity and still achieve SOA:

- People
- Process
- Information
- Reuse
- Connectivity

Prior to SOA, connectivity implied a link between two or more applications or systems. The concept of connectivity is now a service-centric entry point to SOA that is designed to help simplify the 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, the messaging backbone is the foundation for SOA connectivity.

Figure 2-4 on page 16 above shows the IBM representation of Enterprise Service Bus in relation to the reference architecture and shows in Figure 2-5 where WebSphere MQ fits into that architecture as a core component, 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 messaging between applications for many years. The concept of Enterprise Service Bus allows integration of new and existing services using proven technology.

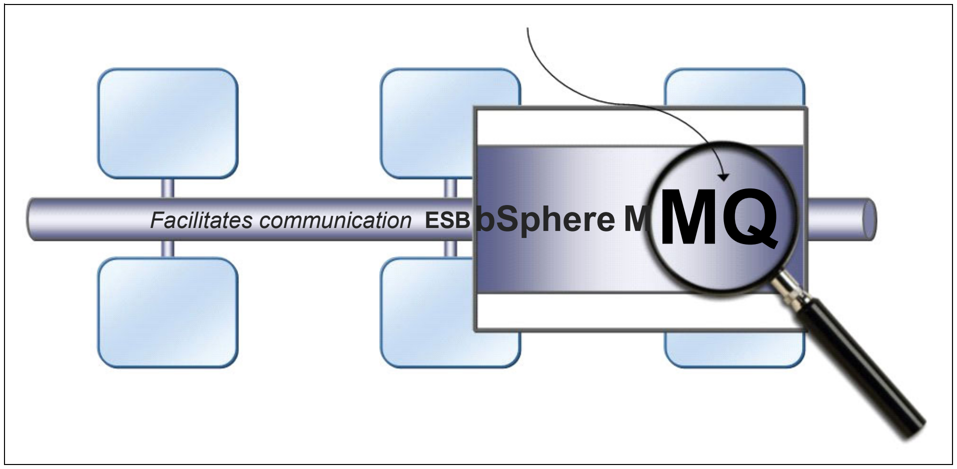


Figure 2-5 WebSphere MQ in relation to reference architecture

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

- Reliable
- Secure
- Time flexible and resilient
- Transactional
- Incremental
- Ubiquitous
- Basis for Enterprise Service Bus

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

3

Introduction to WebSphere MQ

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

This chapter contains three sections:

3.1 Messaging with WebSphere MQ

WebSphere MQ is the market-leading 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 that 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 guaranteed once-only delivery.

Many enhancements have been added to WebSphere MQ during its evolution in the marketplace, 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 fail-over and simplify administration.
- Secure Sockets Layer support: SSL protocol can be used to secure communication between queue managers or MQ Client.
- Diverse platforms: WebSphere MQ supports a wide range of operating system platforms.

Note: To discover and learn about the essential features and capabilities of WebSphere MQ, refer to the IBM Redbooks publication *WebSphere MQWebSphere 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:

- MQ 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 accepting and delivering 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 messaging using both point-to-point and Publish/Subscribe styles.

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

The queue managers within the infrastructure are connected via logical channels over a communications network. 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 that must communicate, whether hosted on the same machine or separate machines, may have originally been designed to do so directly and synchronously. This was a common messaging technique used prior to the introduction of WebSphere MQ.

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

All intercommunication failures that can occur between the two applications must be considered individually by the applications, whether they are on the same machine or on different machines connected by a network. 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 intercommunication failures or recovery.

WebSphere MQ Clients

WebSphere MQ Client is a light-weight component of WebSphere MQ that 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 perform messaging operations with 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 MQ Client can connect to multiple queue managers on different machines.

Important: Because there must be a synchronous communication protocol between the client and the 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 native 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
- Classes in object-oriented languages for accessing data, the queue manager, and its facilities

Many programming languages and styles 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 that are not proprietary to an individual product, such as WebSphere MQ.

Examples of standardized APIs that can be used to access services provided through a WebSphere MQ infrastructure are:

- 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 Enterprise Edition (Java EE) specification.

Note: For information about JMS refer to 2.3, “Java Message Service” on page 13.

Reliability and data integrity

The intercommunication performed across channels between queue managers is tolerant of network communication failures and WebSphere MQ assures 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 messages 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 message storage is based in system memory, so it is possible they can be lost in situations such as network errors, operating system errors, hardware failure, queue manager restart, and internal software failure.

Units of work

Many transactions 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 that 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 any subsequent reply, and updating the information in the database must only complete if all actions are successful.

These actions are considered to be 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 global units of work that are coordinated by other products. For example, actions against a WebSphere MQ infrastructure can be included in global units of work that 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 a 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 Request/Reply messaging interfaces can also be implemented using point-to-point.

The source application needs to know the destination of the message. The queue name is usually enough when alias queues, remote queues, or clustered queue objects are used, but some designs may require the source application to know the name of the remote queue manager.

Publish/Subscribe

WebSphere MQ Publish/Subscribe (Pub/Sub) allows the provider of information to be decoupled from the consumers of that information.

Before a point-to-point application can send 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 that is associated with the destination application. Pub/Sub removes the need for the source application to know anything about the destination application. All it has to do is send information that it wants to share to a known destination topic that is managed and distributed by WebSphere MQ. Similarly, a destination application does not need to know anything about the source of the information that it receives. It only needs to know the topics in which it is interested.

Note: For information about Pub/Sub refer to 2.2, “Introducing Publish/Subscribe” on page 8.

3.1.3 WebSphere MQ distributed messaging

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

The limitations of this topology 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 as the application, providing a fast connection to the infrastructure. MQ channels extend the connectivity by providing transparent asynchronous messaging with applications hosted by other queue managers on remote machines via a communications network.

There are two common types of distributed messaging:

- Hub and Spoke
- MQ Clustering

Hub and Spoke

Applications accessing a service connect to their local queue manager. This is usually via a direct connection to a queue manager that runs on the same machine. A client connection can also be used to a queue manager on the same machine or to a queue manager on a different machine over a fast, reliable network. For instance, an application in a branch office needs to access a service at headquarters. Asynchronous communication occurs through a local queue manager that acts as a spoke to the service application hosted on a hub queue manager.

The machine that 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. This allows 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 that 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 most of the 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 that provide assurance of identity and data privacy for MQ client applications that connect to queue managers via a communications network infrastructure, and also queue manager to queue manager distributed queuing 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 algorithms within the SSL and TLS standards. This ensures 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 Product Information*, available at:

  `http://www.ibm.com/support/docview.wss?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 provided in Part 2, “WebSphere MQ V7.0 enhancements and changes” on page 41.

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 previous versions of WebSphere MQ a separate broker component managed the Publish/Subscribe functionality. It was external to the queue manager 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. It was provided as a SupportPac for WebSphere MQ V5.3 and supplied in the base product for WebSphere MQ V6.0.

WebSphere MQ V7.0 provides a new Pub/Sub engine that is fully integrated into the queue manager and is automatically enabled. The queue manager receives messages from publishers and subscription requests from subscribers for an arbitrary 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 for which publishers provide information. Subscribers interested in information about 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 associates publishers and subscribers. Publishers can publish messages to a topic string and subscribers can subscribe to published messages using a topic string. The topic string can be up to 10,240 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 that defines the security levels of access to topics.

Message selectors

Message selectors can also be used in Pub/Sub messaging. They enable a Pub/Sub application to specify the messages that it is interested in by specifying criteria on message properties.

Distributed Publish/Subscribe

Distributed Pub/Sub enables applications connected to separate queue managers to use Pub/Sub messaging. There are two topologies for distributed Pub/Sub:

- Cluster
- Hierarchical

A *Pub/Sub Cluster* uses MQ Cluster technology that connects queue managers via cluster channels. A MQ Cluster becomes a Pub/Sub Cluster by the definition of at least one clustered topic within the cluster. Although the clustered topic is created on one queue manager, the definition is pushed out to all queue managers in the cluster using the same advertising method as ordinary clustered queues. All publications that are made to the clustered topic are sent to all queue managers in the cluster that have active subscriber applications connected.

A *Hierarchical Pub/Sub* topology is built on queue managers connected via standard distributed message channels or cluster channels. A parent and child relationship is defined to build the hierarchy. The hierarchical topology uses a proxy subscription routing mechanism to deliver messages to subscribers. It can take some time for subscriptions to propagate around all queue managers in the network, and thus publications may not all be received until the proxy subscription has been fully propagated.

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 have the capability for threads and sessions to share a TCP/IP socket. This effectively reduces the number of running TCP/IP sockets, 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 that 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 workload 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 that 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. An 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 introduce 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 that browse the same queue and that are not interested in messages that have been previously browsed by another cooperative dispatcher.

Callback for asynchronous consumers

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

Asynchronous consumer applications must 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 callback 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 callback can be used to retrieve messages from a subscription.

MQSUB has options to create durable or non-durable subscriptions and to specify whether 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
- 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 provider implementation to support new features and to be able to offer functionality similar to 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 about 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 a message asynchronously it can register a callback 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 callback 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 that 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 was 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 the 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 tooling, the Eclipse-based 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, 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 a higher level of serviceability.

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 topics. Testing Pub/Sub is now even easier, with built-in tools to send test publications and receive test subscriptions.

MQSC commands can also be used to manipulate topic objects and subscriptions. The control command **`setmqaut`** is used to administer topic object authority settings.

Managing topics

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

The queue alias type that 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.

On i5/OS®, topics can be managed using the new CL commands.

Managing subscriptions

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

On i5/OS, subscriptions can be managed using the new CL commands.

3.2.7 WebSphere MQ Bridge for HTTP

This feature allows client applications to perform WebSphere MQ messaging using the HTTP protocol. The HTTP bridge feature that was available as SupportPac MA0Y in the previous version has now been incorporated into WebSphere MQ V7.0.

An important aspect of WebSphere MQ is its ubiquity. It is available on a wide range of platforms and operating systems. This feature enables any application with HTTP capability to exchange messages with WebSphere MQ from any platform or language, without the need for a local WebSphere MQ client installation or libraries.

3.2.8 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 in this section.

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. To enable this feature a new queue manager parameter called SCYCASE (Security Profile Case) has been introduced. There are changes to the “REFRESH SECURITY” MQSC command to allow profiles in the new MQ RACF classes to be refreshed.

Using WebSphere MQ Explorer without Client Attach Facility

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 must 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 that 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 that is normally used. The benefits are:

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

3.3 Positioning in WebSphere product family

The WebSphere MQ family of products delivers the universal messaging backbone for service-oriented architecture connectivity. WebSphere MQ provides a flexible and highly scalable transport layer to underpin an Enterprise Service Bus with guaranteed reliable messaging and a choice of qualities of service. The SOA messaging backbone can be enhanced with other IBM products such as WebSphere MQ File Transfer Edition, WebSphere Message Broker, WebSphere Enterprise Service Bus, and WebSphere DataPower® to provide managed data transfer, mediation, transformation, and routing services.

WebSphere MQ integrates virtually any IT system with support for more than 80 platform configurations. It uses standardized interfaces in many programming languages, including support for industry standard JMS, AJAX, Java EE, .NET, CICS, IMS, DB2, and integration with packaged applications. It provides simple access from Web 2.0 user applications to core enterprise back-end applications.

WebSphere MQ can help organizations get more from their IT investments by providing an integration backbone for exchanging messages between applications and Web services. Platform-specific capabilities are exploited to improve service and performance. WebSphere MQ can be extended to provide data protection, enhanced security, and to meet very high volume, low latency requirements.

3.3.1 Foundation for SOA

In a service-oriented architecture, an integration layer, often referred to as an Enterprise Service Bus, enables and optimizes information distribution between an organization’s service components. Underpinning the Enterprise Service Bus layer is a SOA 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 SOA messaging backbone that can help organizations take the first step to SOA.

WebSphere MQ enables Simple Object Access Protocol (SOAP) interactions to flow over the Enterprise Service Bus 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, providing a queuing 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 Enterprise Service Bus in support of SOA.

Note: To learn more about how SOA can integrate the diverse components of complex business environments, refer to the following IBM Web page:

http://www.ibm.com/soa

3.3.2 Enhanced Enterprise Service Bus

An Enterprise Service Bus enables 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. WebSphere MQ provides an ideal basis for implementing an Enterprise Service Bus by delivering a core connectivity layer.

Other members of the WebSphere product family can use a WebSphere MQ-based backbone as the transport layer for application data and inter-operability logic. For example, WebSphere Application Server can benefit from WebSphere MQ capabilities or Enterprise Service Bus implementation.

Note: To learn more about how WebSphere family products can build a solid Enterprise Service Bus backbone, refer to the following IBM Web page:

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

WebSphere MQ and WebSphere Application Server

Java Enterprise Edition (Java EE) compliant application servers, such as WebSphere Application Server, provide a framework in 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 from WebSphere MQ and provides point-to-point and Publish/Subscribe message queuing functionality.

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

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

WebSphere MQ File Transfer Edition V7.0

The newest member of the WebSphere MQ family adds file-specific features to the proven WebSphere MQ transport. WebSphere MQ File Transfer Edition provides a SOA-ready managed file transfer solution that enables the reliable movement of files of any size between IT systems, including to and from ESBs, without requiring any programming. It eliminates the need to depend on File Transfer Protocol (FTP) to provide cross-platform application integration.

WebSphere MQ File Transfer Edition V7.0 delivers support for transfers to and from systems running WebSphere MQ V7.0 and V6.0. At least one WebSphere MQ V7.0 queue manager (supplied with the product) must be deployed within a WebSphere MQ File Transfer Edition network. This queue manager acts as a coordination center for logging audit data and uses WebSphere MQ V7.0 Publish/Subscribe capability to broadcast audit and file transfer status information.

An audit log of file movements enables organizations to demonstrate that business data in files is transferred with integrity from the source to the target file system. Graphical configuration tooling, integrated with WebSphere MQ Explorer, enables quick and easy definition and management of automated and manuals transfers. Transfers may also be scripted or performed programmatically. For further information, refer to the following IBM Web site:

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

Part 2

WebSphere MQ V7.0 enhancements and changes

This part describes the new features, enhancements, and changes in WebSphere MQ V7.0 and provides information about installation tasks and the migration path from 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 providing Pub/Sub, Callback, Selectors, and message property capabilities
- 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 queue manager. This chapter starts with a brief introduction to the concepts of WebSphere MQ Publish/Subscribe and then gives a conceptual overview of the Publish/Subscribe engine in V7.0.

This chapter contains the following sections:

4.1 Publishing and subscribing in WebSphere MQ

Applications that produce information about a particular subject are referred to as publishers. Applications that consume this information are referred to as subscribers. The information and subjects are managed by WebSphere MQ Publish/Subscribe (Pub/Sub). The subject is referred to as a topic. The information equates to WebSphere MQ messages.

Subscribing applications register their intention to receive information from particular topics with WebSphere MQ Pub/Sub. Publishing applications then send information about topics to WebSphere MQ Pub/Sub. The management and distribution of the information to registered subscribers is the responsibility of WebSphere MQ Pub/Sub. This decoupling of publisher and subscriber applications allows 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 would be created for each queue manager, and it used the same name as the queue manager. This broker used WebSphere MQ facilities to manage interactions between publishing and subscribing applications.

WebSphere MQ V7.0 provides a new Publish/Subscribe engine that is integrated into the queue manager. This is a major enhancement because the queue manager now internally manages all the Publish/Subscribe functionality. The queue manager receives messages from publishers and subscription requests from subscribers for a range of topics, which is responsible for queuing and routing these messages to the target subscribers. Section 12.4.5, “Publish/Subscribe engine” on page 250, provides general information about making the WebSphere MQ V7.0 and V6 Publish/Subscribe engines interoperate and provides an overview of the migration path to the new native engine. Upcoming sections of this chapter describe the major Publish/Subscribe enhancements in WebSphere MQ V7.0 in greater detail.

The fact that Publish/Subscribe functionality is now integrated into the queue manager simplifies application programming a great deal. In previous versions of WebSphere MQ, Publish/Subscribe was not a native part of the MQI. Applications were required to communicate via a queued interface to a broker process running outside the queue manager. New MQI API calls have been introduced in WebSphere MQ V7.0 to support the native Publish/Subscribe feature. Chapter 6, “Message Queue Interface extensions” on page 85, provides details of these new API calls and further details on the types of publications and subscriptions available in WebSphere MQ V7.0. The WebSphere MQ V7.0 Information Center provides a new manual, *Publish/Subscribe User's Guide*, which expands on the information in this book. It is available at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

4.2.1 Topics in WebSphere MQ V7.0

A topic refers to the subject on which publishers provide information. Subscribers interested in information about 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 provides the logical association between publishers and subscribers. Publishers can publish to a topic string and subscribers can subscribe to the publications using the topic string. A topic string can be up to 10,240 characters long and is case sensitive. A new data-type called the variable-length string (MQCHARV) has been introduced in WebSphere MQ in order to support the long string requirement.

The structure and semantics of the topic string are controlled by the slash (/). For example, there can be a high-level topic called *deli* to represent a delicatessen, which might be divided into separate sub-topics relating to different categories of products that the deli sells, and further layers of sub-topics beneath that to further qualify the product. Example 4-1 shows some typical topic strings.

Example 4-1 Topic string examples

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

Topic strings imply a sense of hierarchy in the topic structure. The hierarchy is represented as the topic tree, as depicted in Figure 4-1. The topic tree has a root node that corresponds to the topic object SYSTEM.BASE.TOPIC.

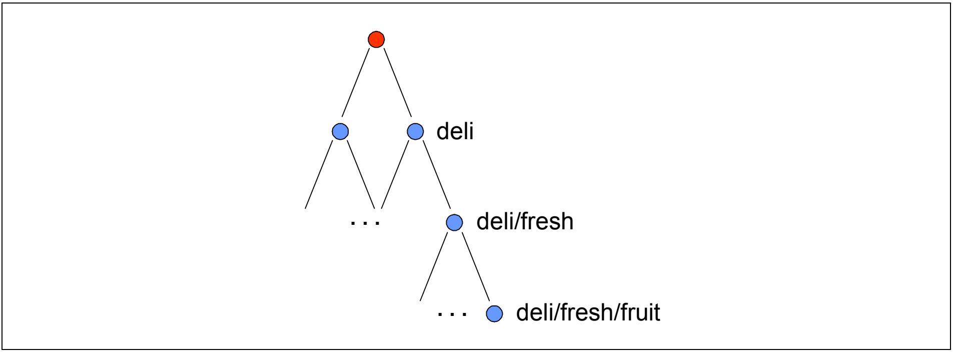


Figure 4-1 Topic tree

Topic strings support wildcard characters. Subscribers can use two wildcards, hash (#) and plus (+), to subscribe to a range of topics. Both provide methods of 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. We recommend not using these characters in topic strings when publishing. Chapter 9, “Publish/Subscribe management” on page 197, provides further details on how to achieve this.

Note: For compatibility with previous versions, an alternative wildcard scheme is available. The wildcards question mark (?) and asterisk (*) can be used in place of hash (#) and plus (+).

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

Topic objects are administrative objects that are defined in WebSphere MQ. Administrative topic objects allow attributes to be defined for certain portions of the topic tree. For example, authority checking may be set up on a topic object to control whether that portion of the tree can be published or subscribed, and by who. The predefined base topic called SYSTEM.BASE.TOPIC represents the root of the topic tree. It is not necessary to define any other topic objects if security and topic attributes are the same for the entire topic tree, since these settings are inherited from the parent topic object that exists higher up the tree. Thus, it is possible to get a Publish/Subscribe application up and running without defining any topic objects at all.

Topic strings are used to match information from a publisher to subscribers who are interested in that information. Topic strings do not have to be predefined. They come into existence dynamically when subscribing and publishing applications use them. Consider a publisher application publishing to a topic string called deli/fresh/fruit and no administrative topic objects have been defined at this time. The nodes on the corresponding tree, as shown in Figure 4-1 on page 46, are referred to as non-administrative topics. It is possible to define an administrative topic object for any node on this sub-tree (for example, /fresh/fruit) only if there is a need to associate specific attributes' settings with that particular node that are not the same settings as inherited from the parent node.

Furthermore, defining an administrative object on /fresh/fruit may not be necessary if there is an administrative object defined for deli/fresh that already has these specific non-default attributes defined. 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.

Although subscribers can 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 also allows administrators to shield or demarcate portions of the topic tree by defining an administrative topic at the highest point in the topic tree up to which an application needs to know, and then create non-administrative nodes below that point by using topic strings. For example, if there is an administrative topic object called DELI.FRESH defined for the topic string deli/fresh, then publishers and subscribers can use this object name in conjunction with the name of the sub-topic type that they are interested in as the topic string, for example, fruit. This has the same effect as publishing or subscribing to deli/fresh/fruit. Chapter 9, "Publish/Subscribe management" on page 197, provides further details on how to achieve this.

4.2.3 Topic alias

An alias queue is a WebSphere MQ object that provides a level of indirection or reference to another queue object. The queue object name resulting from the resolution of an alias 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 for z/OS).

WebSphere MQ V7.0 introduces an extension to the alias queue object that allows it to be resolved to the new topic object. This is useful for migrating point-to-point messaging applications to the Publish/Subscribe model. A traditional point-to-point application that puts messages into WebSphere MQ can operate as a publisher without any code changes by utilizing an alias queue that resolves to a topic object. This is implemented administratively by defining a topic object that maps to an appropriate topic string on which the messages are to be published. The original local queue is deleted and replaced by an alias queue of the same name that resolves to the topic object.

Also note that a point-to-point application that gets messages from WebSphere MQ can operate as a subscriber without any code changes by defining an administrative subscription to a topic. This procedure is described in 9.2, “Managing subscriptions” on page 216. Each point-to-point application requires a local queue to be defined for it to get the published messages. If this local queue is on the same queue manager as a partner point-to-point application that has been converted to be a publisher by using a topic alias, the local queue must have a different name from the alias queue object used by the publisher.

The introduction of the topic alias and administrative subscription allows point-to-point applications to enjoy the versatility 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 the queue that are then retrieved by a single message consumer. By using the approach just described, it is possible for multiple point-to-point applications to generate statistics, and for multiple point-to-point applications interested in processing the statistics messages to subscribe to the topic and consume the messages.

Administration of the queue alias object for topics is described in 8.2.2, “Queue object parameters” on page 180.

4.2.4 Topic security

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

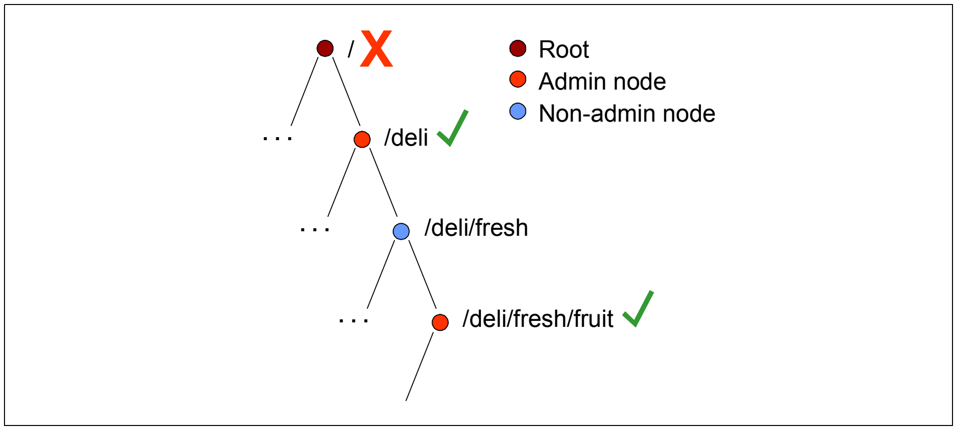


Figure 4-2 Delegation of topic security attributes

For example, if an application wants to subscribe to the topic string /deli/fresh/fruit, and /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 MQSC commands as well as the MQ explorer. Chapter 9, “Publish/Subscribe management” on page 197, provides further details on how to achieve this.

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. 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) it may still be eligible for selection.

Furthermore, WebSphere MQ V7.0 now provides native support for message properties and allows applications to set and get properties in a message with arbitrary names and values, whether using JMS or the MQI. Details on using message properties in MQ can be found under Message Properties and Handles in 6.2, “Message properties” on page 89.

The JMS implementation in WebSphere MQ V6 performed message selection on the client side. Messages were browsed sequentially from a queue until the client finds one that matches the selection criteria, at which point a destructive get is performed on that message. The WebSphere MQ V7.0 implementation makes message selection much more efficient by introducing selection processing on the queue manager (server side). 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. Message selection is now also supported for MQ API applications.

Syntax for WebSphere MQ V7.0 selector strings

A MQ message selector is a variable-length string and its 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. Parentheses can be used to change this order.

Literals

String literals in a selector string are specified using a single quotation mark, for example, ‘MQ’ or ‘WebSphere MQ’. Byte literals, which are one or more pairs of hex characters, must be specified in double quotation marks, for example, “0XD43A”. Boolean literals can take values TRUE and FALSE. All types of numeric literals, including decimal, hexadecimal, and octal numbers, do not need to be specified in quotation marks.

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 can be used in a selector string:

- Arithmetic operators
- Logical operators
- Comparison operators

Valid arithmetic operators include:

- + (both unary and binary plus)
- - (both unary and binary plus)
- * (multiplication)
- / (division)

The precedence order followed for these operators is:

1. +
2. -
3. *
4. /

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. The precedence order for logical operators is:

1. NOT
2. AND
3. OR

There are two other operators:

- LIKE, which is used for pattern matching
- BETWEEN, which is used in arithmetic comparisons

Traditional type conversion rules do not hold true for selectors. For example, if a message had a message property as a string value and a selector is used 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 that does not exist, the property is assumed to have the value of NULL or Unknown. 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 that map to property names or MQMD field names may be used as valid identifiers in a selection string. WebSphere MQ 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
- JMSType

Values can be empty, and if so, are treated as a NULLvalue.

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 WebSphere MQ JMS implementation. Rules governing this mapping are discussed in the section “Mapping JMS messages onto WebSphere MQ messages” of the *Using Java* manual, which is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

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 and 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 provides examples of the evaluation of selectors.

Table 4-1 Selector string examples

Type	Selector	Evaluates to	Governing rules
Selectors using Byte Literals. 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. ► The byte order is not affected by a Big Endian or Small Endian machine architecture. ► 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 -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 are supported and are internally stored as signed long integers. ► Hex numbers begin with a zero followed by an upper/lowercase 'x'. The remainder of the literal should contain n valid hex characters, where n > 1 (for example, 0xA, 0xAF, 0x2020, 0XE). ► Octal numbers begin with a leading zero that is followed by n digits in the range 0–7, where n >= 1 (for example, 0177, 0173). Note that a leading zero that is followed by one or more digits is always interpreted as being the start of an octal number, meaning that it is not possible to represent a zero-prefixed decimal number in this way. For example, '09' would return a syntax error because 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 > 2500 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 allows applications connected to separate queue managers to communicate via publish and subscribe. This can be achieved by two topologies:

- Clusters
- Hierarchical

4.4.1 Pub/Sub Cluster topology

A Pub/Sub Cluster uses MQ Cluster technology that connects queue managers together via cluster channels. A MQ Cluster becomes a Pub/Sub Cluster by the definition of at least one clustered topic within the cluster. Although the clustered topic is created on one queue manager, the definition is pushed out to all queue managers in the cluster using the same advertising method as ordinary clustered queues. All publications that are made to the clustered topic are sent to all queue managers in the cluster that have active subscriber applications connected, as illustrated in Figure 4-3.

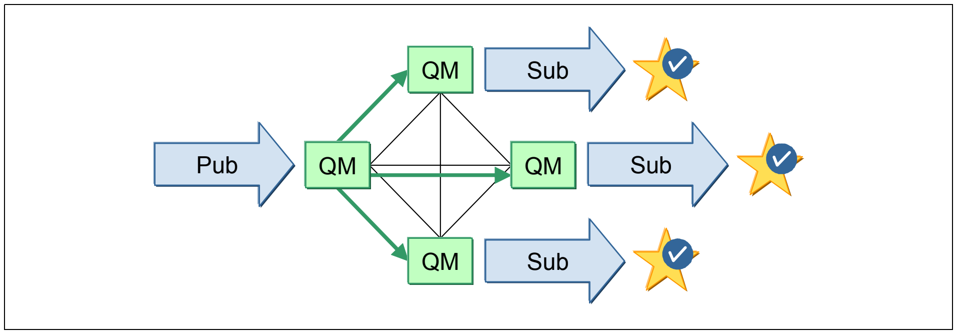


Figure 4-3 Publish/Subscribe clusters

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 application's connected queue manager to all other queue managers in the MQ Cluster on which the clustered topic object is defined. If a queue manager on which the clustered topic object is defined becomes unavailable, the subscription will remain in place for up to 30 days, so that normal Pub/Sub activity is restored when the queue manager becomes available.

Publishing to clustered topics

Publications in a Publish/Subscribe cluster are sent to those queue managers for which proxy subscriptions have been received.

Note: A single queue manager can be a member of more than one Pub/Sub Cluster. This would be done to create a gateway between two clusters so that messages originating in one Pub/Sub Cluster can be routed to another Pub/Sub Cluster. Although a 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 topology must be used.

4.4.2 Pub/Sub hierarchical topology

A Pub/Sub hierarchical topology is built on queue managers connected via standard distributed message channels or cluster channels. A parent and child relationship is defined to build the hierarchy, as illustrated in Figure 4-4. The hierarchical topology uses a proxy subscription routing mechanism to deliver messages to subscribers. It can take some time for subscriptions to propagate around all queue managers in the network, and thus publications may not all be received until the proxy subscription has been fully propagated.

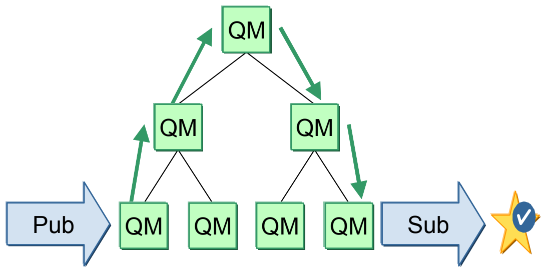


Figure 4-4 Hierarchical distributed queue managers

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, we strongly recommend 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 looping 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 completely circulated around a loop, so the queue manager discards the message and reports it on the error log.

4.4.4 Scope of publications and subscriptions in Distributed Pub/Sub

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 the publication or subscription is delivered or propagated.

The scope of publications can be controlled administratively using the PUBSCOPE parameter on a topic object. The parameter 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.

Similarly, the scope of subscriptions can be controlled administratively using the SUBSCOPE parameter on a topic object. The parameter can be set to one of the following values:

- QMGR: The subscription is not propagated to directly connected queue managers, and only receives publications from local publishers.
- ALL: The subscription is propagated directly to connected queue managers, and receives 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. However, the definition of the clustered topic object is still shared with other queue managers in the cluster.

5

WebSphere MQ Client enhancements

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

MQ Client is a set of libraries or classes that 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 that can run a WebSphere MQ queue manager.

Refer to the manual *WebSphere MQ Clients* for further details on platform requirements, installation, configuration, administration, and programming using the client. This manual is available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

This chapter contains the following sections:

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 protocol that MQ uses over the TCP/IP 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 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 that is prefixed by an asterisk (*). This directs MQ to attempt connection using an alphabetic list of CLNTCONN type channels defined in 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 that was selected by a successful call to MQCONNX. The channel name can also be passed to MQCONNX to override the normal algorithm that 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 4 MB to 100 MB.

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 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 that have been incorporated into the general MQI. They are described in detail in Chapter 6, “Message Queue Interface extensions” on page 85:

- Publish/Subscribe (The principles are explained in Chapter 4, “Publish/Subscribe integration” on page 43.)
- Getting and setting message properties using Message Handles=.
- Message selectors.
- Cooperative browsing using Message Tokens.
- Asynchronous Consume and Event notification using Callback.

The following sections contain detailed descriptions of enhancements that are targeted specifically at WebSphere MQ Client. There is discussion of 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 Chapter 17, “Scenario: News using client” on page 325, for examples of WebSphere MQ Client programs that make appropriate use of read ahead and asynchronous put. Other components of the scenario also use the client but do not necessarily take advantage of the enhancements in WebSphere MQ V7.0.

5.2 Full duplex channels, heartbeat, 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 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. Heartbeats are short flows of information that are sent at a regular interval. Their only purpose is to confirm that the session is still active. Heartbeats can now be performed from both the client end and the queue manager end at a negotiated rate that is based on the HBINT parameter of both 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 heartbeat 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* for further details of this feature. This manual is available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

Figure 5-1 and Figure 5-2 on page 64 show the difference between half duplex and full duplex operation of a WebSphere MQ Client channel.

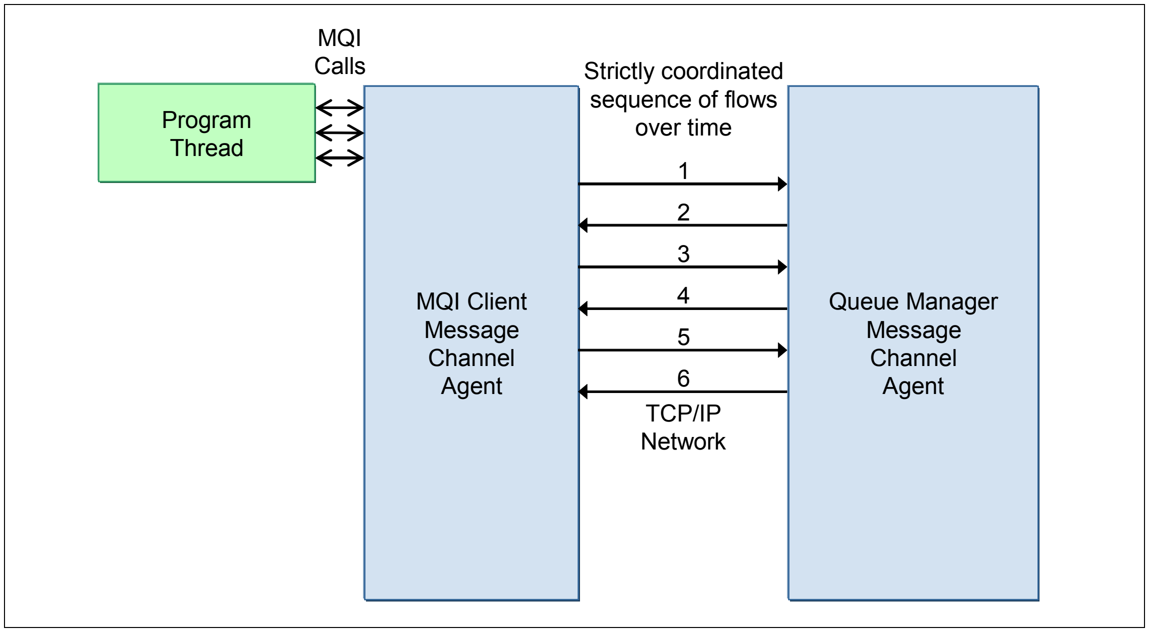

MQI Calls
Program Thread
Strictly coordinated sequence of flows over time
1
2
3
4
5
6
TCP/IP Network
MQI Client Message Channel Agent
Queue Manager Message Channel Agent

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

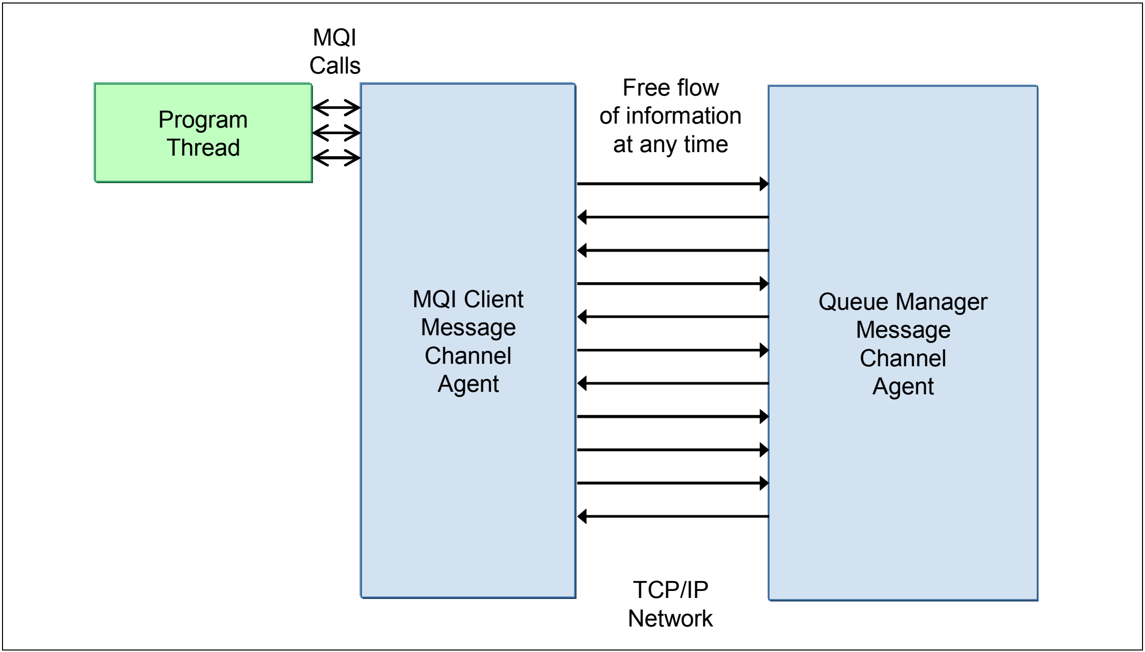

MQI Calls
Program Thread
Free flow of information at any time
MQI Client Message Channel Agent
Queue Manager Message Channel Agent
TCP/IP Network

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 share a 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 socket session. There is also only one pair of heartbeat flows and one flow to stop the socket.

Multi-threaded client programs running on previous versions of WebSphere MQ client or server use a separate channel instance for each thread. For a WebSphere MQ V7.0 client connected to a WebSphere MQ V7.0 queue manager, the conversation sharing feature reduces the number of running TCP/IP sockets on the queue manager, resulting in more efficient use of resources.

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

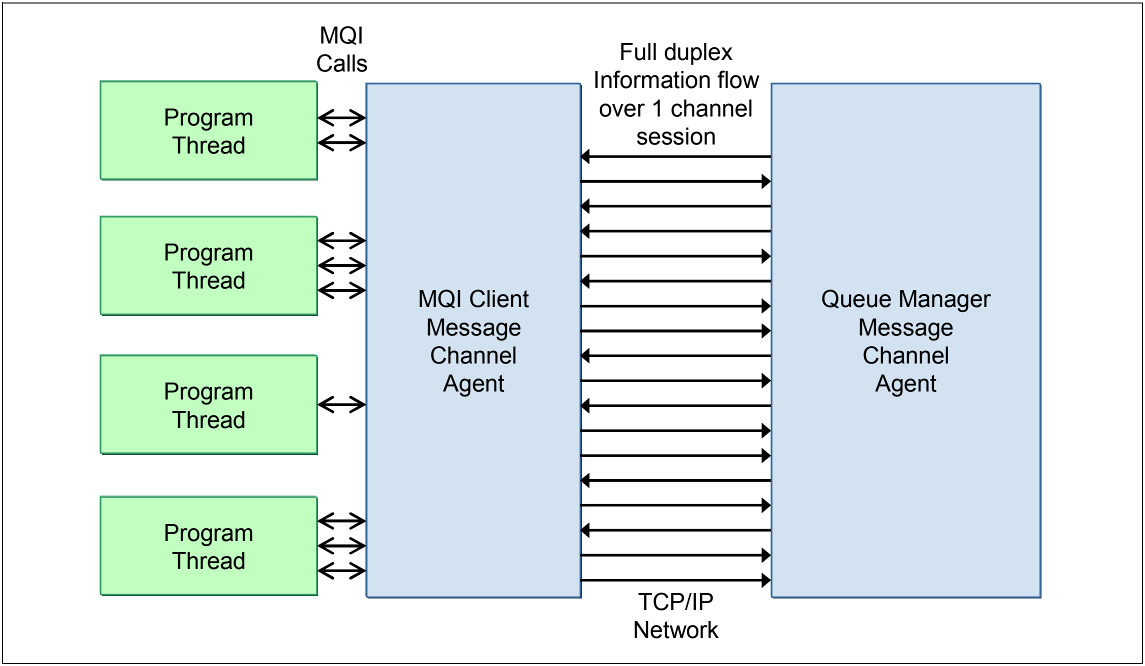


Figure 5-3 Multi-thread conversation sharing in WebSphere MQ V7.0

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 that is connected to MQ 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 connected threads in a single client program that 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 connected threads that can share a single instance.

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

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 is set to the default value of 10. This default is also used 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.

CLNTCONN-type channels can be defined using a Client Channel Definition Table (CCDT) file, the MQSERVER environment variable, the MQCONNX API call, and the Java/JMS classes. When a CLNTCONN channel is used with WebSphere MQ V7.0 client libraries the SHARECNV parameter takes the default value of 10.

On the MQCONNX call, the default can be changed by setting the SharingConversations field of the MQCD data type. In Java, the default can be changed by setting the sharingConversations field of the MQEnvironment and MQChannelDefinition classes.

To change the default on a Client Channel Definition Table file that was created prior to WebSphere MQ V7.0, the CLNTCONN channels defined in the file need to be manually altered using a WebSphere MQ V7.0 queue manager to set the SHARECNV parameter. Refer to the sections on environment variable MQCHLTAB and the Client Channel Definition Table in the manual *WebSphere MQ Clients* for further details. This manual is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

5.3.2 MQCONNX options for conversation sharing

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 related to conversation sharing:

- 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 that there is no conversation sharing with any other thread of the client program.

Example 5-1 contains code in the C language that sets a conversation sharing option 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:

- CURSHCNV is a new field that reports the number of conversations that are currently connected to the queue manager on the channel instance.
- MAXSHCNV is a new field that reports the maximum number of conversations that 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 that 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 that are sharing the channel instance.
- LSTMSGDA and LSTMSGTI represent the date and time of the most recent MQI call made by any thread that is sharing the channel instance.

Refer to Chapter 8, "Administration enhancements" on page 153, 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 that 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*. This manual is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

5.4 Read ahead

A new feature of the WebSphere MQ V7.0 client can provide an increase in throughput for client programs that get a sequence of non-persistent messages. A good example is an enquiry business function that results in a back-end application generating a number of reply messages on a queue that 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 that 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 is quite acceptable in many circumstances.

If the read ahead function encounters a persistent message it stops reading messages into the client memory until the persistent message has been received 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 for 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 Callback. Refer to Chapter 6, “Message Queue Interface extensions” on page 85, 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 that 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 that 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 that 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 that 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, “Message Queue Interface extensions” on page 85, for details.

5.4.2 MQGET considerations

Messages can be requested by the client program with matching Message Id 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 that are requested by the program.

The first request to get a message from a queue determines whether messages are being browsed or got destructively from the queue manager by the read ahead feature. 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 that has read ahead enabled there may be non-persistent messages that 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 TYPE(HANDLE) 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 153, 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 that 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 sync pointing 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 sync point 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 sync point to take advantage of asynchronous put, as persistent messages that are put without sync point 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 correct circumstances. Figures 5-4 and 5-5 show the interactions required without asynchronous put and then with it.

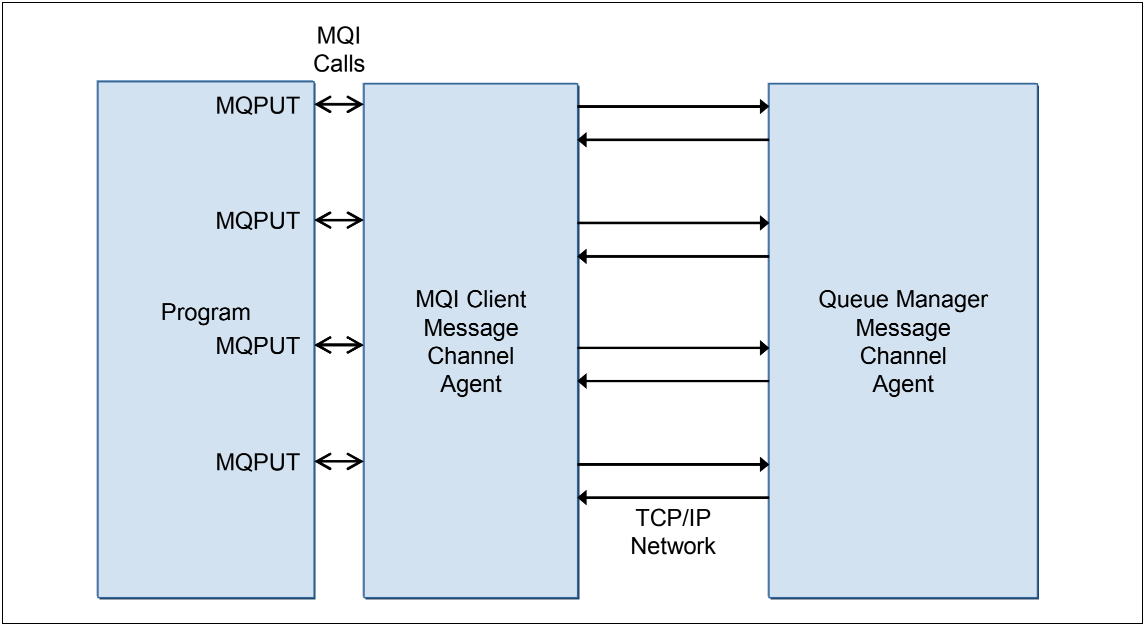


Figure 5-4 Client putting messages without asynchronous put

An example of using this feature is a client program that 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 obtain the status information.

If a unit of work was in progress and persistent messages were put with sync point, a call to MQCMIT that is 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 MQ Message Descriptor (MQMD) and output fields in the put options are not updated in the client program memory. This is because they are part of the response from the queue manager, which has not yet been received. The

MessageId field is updated in the MQMD because the unique value is assigned by the client.

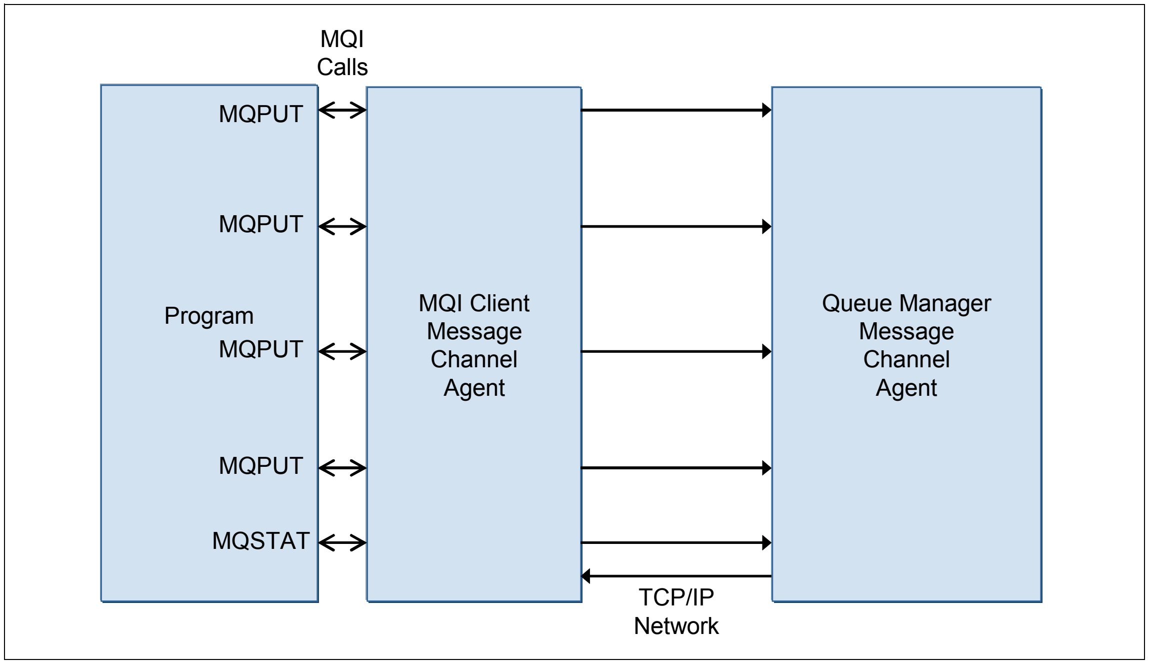

MQI Calls
Program
MQPUT
MQPUT
MQPUT
MQPUT
MQSTAT
MQI Client Message Channel Agent
Queue Manager Message Channel Agent
TCP/IP Network

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, a call to get a message with a specific Correlation ID does not return to the program until all prior asynchronous puts on that queue have been fully actioned by the queue manager.

5.5.1 MQPUT and MQPUT1 options for asynchronous put

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

- MQPMO_RESPONSE_AS_QDEF is the default if none of the options are 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 sync point 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 that is detected prior to passing the message to the queue manager. The program may later call MQSTAT to obtain the actual status response. If a persistent message was put in sync point, the status response from a later call to MQCMIT is sufficient to indicate whether all the messages in the unit of work were successfully put.
- There is a fourth option that only applies to a topic object that 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 that 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 that 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 85.

Example 5-2 contains code in the C language that puts some messages asynchronously and then uses MQSTAT to retrieve the status information.

Example 5-2 C code: 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( “Failed to obtain status, Completion Code %d, ”
         “Reason Code %d.\n”, CompCode, ReasCode );
      ... (take appropriate action)
      }
   else
      {
      printf( “%d puts succeeded, %d had warnings, %d failed.\n”,
         StatInfo.PutSuccessCount, StatInfo.PutWarningCount,
         StatInfo.PutFailureCount );
      if( StatInfo.CompCode != MQCC_OK )
         {
         printf( “The first error was Completion Code %d, ”
            “Reason Code %d.\n”, StatInfo.CompCode, StatInfo.Reason );
      ... (take appropriate action)
         }
      else
         printf( “No errors.\n” );
      }
   ...
```

5.6 Instance limits on SVRCONN channels

The SVRCONN type channel has been enhanced in WebSphere MQ V7.0 to add two new parameters that 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 that 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 that use the queue manager. Therefore it is very useful to be able to limit the distribution of maximum running channels on individual SVRCONN channels, leaving capacity for other types of channels to start when they need to.

Refer to the manual *WebSphere MQ System Administration* for further details on the MaxChannels, MaxActiveChannels, MAXCHL, and MAXTCP parameters that set the absolute maximums for the queue manager. This manual is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

If the maximum was not specified when the queue manager was created, or if it was 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 that must 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 that can be run by all client programs connecting to the queue manager via the channel. A client program that uses 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 that 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 that can be run by all client programs connecting to the queue manager via the channel from a unique network address. A client program that uses 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 that 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 that 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 five 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 153, 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 that is prefixed by an asterisk (*):

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

This feature is demonstrated in the section “Examples of using MQCONN calls” of the manual *WebSphere MQ Clients,* which is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

The asterisk instructs MQ to attempt connection using a set of channel names that are defined in a Client Channel Definition Table. The algorithm builds an alphabetic list of channel names that 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 *backup* queue manager that 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 workload 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 that the channel has no weighting. A value in the range 1 to 99 indicates a weighting that is relative to other channels that have a non-zero weighting and also match the queue manager name.

The new algorithm first considers the matching channels that 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 that 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 in the list, it is moved to the end of the list, and the next one in 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”. That 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 that 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 are used, provided that 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 that are used by other active connections in the process.

5.8 Reconnecting via a previously used channel

A client program can make a connection request to a queue manager name that is blank, prefixed by an asterisk (*). MQ may then establish the connection by one of several possible channels in the Client Channel Definition Table. The actual channel name and connected queue manager name are 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 that 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 that is used to process CCDT files.

In WebSphere MQ V7.0 there are two new options related to reconnection:

- MQCNO_CD_FOR_OUTPUT_ONLY
- MQCNO_USE_CD_SELECTION

Example 5-3 contains code in the C language that 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 code given in Example 5-4 demonstrates this as a continuation of the code in Example 5-2 on page 75.

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 are used (transport type, host name, port number). Otherwise, the configuration details from the CCDT are used.

5.9 Max 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 or a programmatic method such as MQCONNX in the MQI to define the channel.

Only the channel name, transport protocol, and 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 100 MB (104,857,600 bytes). This has been increased from the 4 MB (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 4 MB, 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 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 153.

5.11 Using MQ Explorer without a 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 Client Attach Facility (CAF) license for WebSphere MQ V7.0 on z/OS. Up to five 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 197, for further details on enabling this capability.

5.12 Compatibility

Version 6.0 and 7.0 WebSphere MQ Clients can connect to Versions 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 that 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 feature that is available for a Version 6.0 client connected to a version 7.0 queue manager is instance limits on SVRCONN channels.

CLNTCONN channel definitions in a Client Channel Definition Table file that has 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.

CCDT files required for use with previous versions of WebSphere MQ Clients can be maintained using a matching version of WebSphere MQ queue manager. As an alternative to keeping earlier versions of queue managers just for this purpose, IBM SupportPac MO72 MQSC Client for WebSphere MQ 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`

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*, which is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

This chapter covers the following new functionality in WebSphere MQ V7.0 and includes the following sections:

- 6.1, “Variable-length strings” on page 86
- 6.2, “Message properties” on page 89
- 6.6, “Put action indicators” on page 133
- 6.3, “Message browsing” on page 105
- 6.4, “Callback for asynchronous consumers” on page 108
- 6.5, “Publish/Subscribe” on page 121
- 6.7, “Message selectors” on page 134
- 6.8, “Other MQI considerations” on page 135

6.1 Variable-length strings

Variable-length strings are used for some of the new features in WebSphere MQ V7.0. These are represented as a new data type called MQCHARV. Topic strings, subscriber user data, and selection strings are some of the items represented as MQCHARV.

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

There are two methods to specify the location of a variable-length string, using a pointer or using an offset. The offset method is required for some programming languages like COBOL in some environments that do not support the use of pointers.

There are also two methods to specify the length of the string, as the actual string length in a MQLONG variable, or to specify that the string is null terminated.

Applications must allocate a buffer to receive or to send variable-length strings to or from a WebSphere MQ V7.0 queue manager. If a buffer is not specified, or is too small 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

Table 6-1 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. This is null when VSOffset is used.
VSOffset	MQLONG	0	Offset to string buffer. This is zero when VSPtr is used.
VSBufSize	MQLONG	MQVS_USE_VSLENGTH	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

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 VSPtr to the location of the string buffer.
3. Set VSOffset to zero.
4. If the call is providing a string as input to a Message Queue Interface call:
   - Place the string in the buffer and store the length in VSLength.

   OR

   - Place the null terminated string in the buffer and set VSLength to MQVS_NULL_TERMINATED.
   - The coded character set ID of the string may be stored in VSCCSID. The default is MQCCSI_APPL.
5. If the call is returning a string to the application program from a Message Queue Interface call:
   - The string will be placed in the buffer. No null termination character is put in the buffer.
   - The actual string length will be placed in VSLength.
   - The desired coded character set ID of the string may be stored in VSCCSID. The default is MQCCSI_APPL. The queue manager tries to convert the string to the required CCSID, but if it is not possible it returns the actual CCSID of the returned string in VSCCSI.

6.1.3 MQCHARV with offset

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 to the beginning of the string buffer. The offset can be a positive or negative value.

   Note: The offset is calculated from the beginning of the structure that is being used as the parameter to the Message Queue Interface call. For example, when used in an MQOD to describe a SelectionString the offset is from the beginning of the MQOD structure and not from the SelectionString MQCHARV.

4. If the call is providing a string as input to a Message Queue Interface call do one of the following steps:
   - Place the string in the buffer and store the length in VSLength.
   - Place the null terminated string in the buffer and set VSLength to MQVS_NULL_TERMINATED. The coded character set ID of the string may be stored in VSCCSID. The default is MQCCSI_APPL.
5. If the call is returning a string to the application program from a Message Queue Interface call:
   - The string will be placed in the buffer. No null termination character is put in the buffer.
   - The actual string length will be placed in VSLength.
   - The desired coded character set ID of the string may be stored in VSCCSID. The default is MQCCSI_APPL. The queue manager tries to convert the string to the required CCSIDl, but if it is not possible it returns the actual CCSID of the returned string in VSCCSI.

Figure 6-1 shows the relationships between VSPtr, VSOffset, VSBufSize, and VSLength in a MQCHARV data structure.

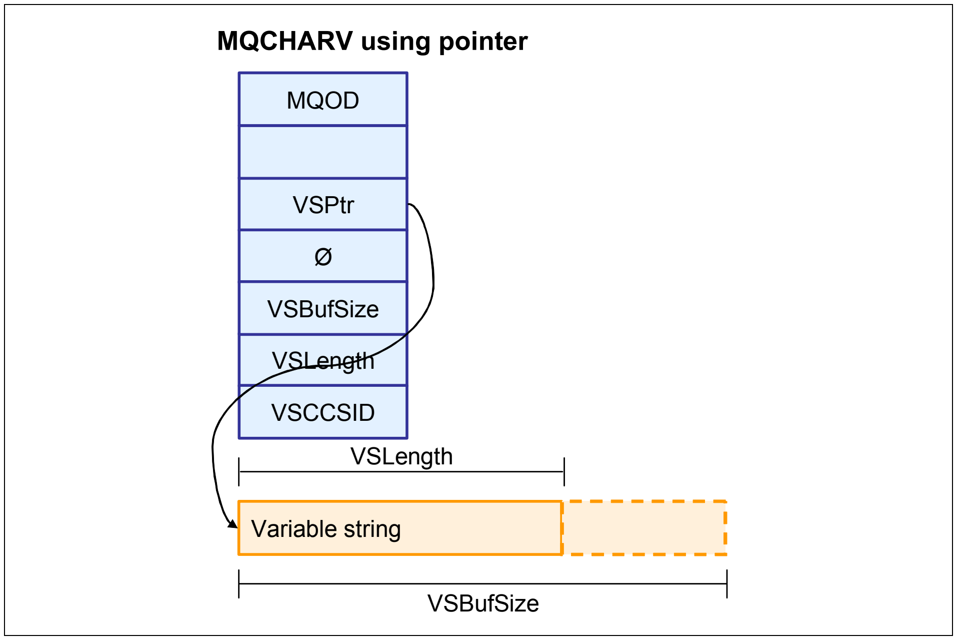


Figure 6-1   MQCHARV

6.1.4 Null terminated strings

If the call is providing a string as input to a Message Queue Interface call then a null terminated string may be used. The string (together with the null character) is placed in the buffer and the value MQVS_NULL_TERMINATED is placed in VSLength.

If the call is returning a string to the application program from a Message Queue Interface call then a null terminated string is *never* used. The actual length is placed in VSLength.

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 must 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 a new feature of WebSphere MQ V7.0. They are name-value pairs that are carried alongside the message body. Java Message Service (JMS) introduced the concept of message properties, and now they are supported directly in WebSphere MQ V7.0 as part of the MQI.

Message properties are used by message selectors to filter subscriptions 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. They also give MQI Applications direct access to message properties set by a JMS application.

From WebSphere MQ V7.0 onward 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 function calls described in this section.

6.2.1 Message handles

Message handles are new in WebSphere MQ V7.0. They are references to a message and are used to access and set message properties.

There are new Message Queue Interface 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 gives access 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.

6.2.2 Set, inquire, and delete message properties

There are new Message Queue Interface function calls 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 be used to create or to inquire message properties. This message handle will be associated with the connection handle (Hconn) used in the call. Any resources used in the message handle will be freed when the application disconnects.

Syntax

The MQCRTMH function syntax is:

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

Parameters

Table 6-2 lists the parameters that are used by MQCRTMH.

Table 6-2 MQCRTMH parameters

Parameter name	Data type	Input/output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the queue manager
CrtMsgHOpts	MQCMHO	Input	Options to control how message handles are created
Hmsg	MQHMSG	Output	Created message handle
CompCode	MQLONG	Output	Completion code
Reason	MQLONG	Output	Reason code

Example

Table 6-3 on page 92 shows how to create a message handle for 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);

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

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

Note: Message handles created by using the MQHC_UNASSOCIATED_HCONN as a connection handle are not associated with a specific connection. Such message handles can be used on MQGET/MQPUT function calls for any connection. However, the resources used must be explicitly freed with a MQ DLTMH call, as they are not released when the application disconnects.

6.2.4 MQDLTMH delete message handle

This function call deletes a message handle and releases any associated resources.

Syntax

The MQDLTMH function syntax is:

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

Parameters

Table 6-3 lists the MQDLTMH parameters.

Table 6-3 MQDLTMH parameters

Parameter name	Data type	Input/output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the queue manager. It must be the same connection used when the handle was created. If the handle was un-associated to a connection then this must be MQHC_UNASSOCIATED_HCONN.
Hmsg	MQHMSG	Input/Output	Handle to be deleted. 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

Example 6-2 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

The MQSETMP function syntax is:

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

Parameters

Table 6-4 lists 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	Used to define attributes of a property. This parameter is intended for advanced uses of message properties.
Type	MQLONG	Input	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

Example 6-3 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 the 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

The MQINQMP function syntax is:

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

Parameters

Table 6-5 lists 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	Used to define attributes of a property. This parameter is intended for advanced uses of message properties. Most applications will not need to set these attributes.
Type	MQLONG	Input/Output	Data type of the property, such as 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.
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 to return the entire value.
CompCode	MQLONG	Output	Completion code.
Reason	MQLONG	Output	Reason Code.

Example

Example 6-4 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

The MQDLTMP function syntax is:

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

Parameters

Table 6-6 lists 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 the 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

Example 6-5 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

The MQBUFMH function syntax is:

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

Parameters

Table 6-7 lists the MQBUFMH 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	Length in bytes of the buffer with the properties removed.
CompCode	MQLONG	Output	Completion code.
Reason	MQLONG	Output	Reason Code.

Example

Example 6-6 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 converts 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

The MQMHBUF function syntax is:

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

Parameters

Table 6-8 lists 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	Name of the 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	Area to contain the message with the message properties in the MQRFH2.
DataLength	MQLONG	Output	Length of the message returned in buffer including the properties.
CompCode	MQLONG	Output	Completion code.
Reason	MQLONG	Output	Reason Code.

Example

Example 6-7 lists 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 commonly 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 a 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 with connection handles. Therefore, two dispatchers browsing the same queue browse the same messages.

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

- Support of browsing with mark to allow browsing of only messages that have not yet been marked by this application.
- Support of browsing with cooperative mark to allow multiple applications to browse the same queue and see only messages that no other application has yet cooperatively marked.
- Support of message tokens for distributed queue managers as well as z/OS queue managers. These message tokens provide an efficient mechanism for a dispatching application to pass a reference to a particular message to another application.

6.3.1 Message tokens

The message token is a unique identification that remains with a message until it is permanently removed from the queue or the queue manager is restarted. The 16-byte identifier is generated by the queue manager and returned after MQPUT or MQGET function calls.

This new feature allows applications to do efficient 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 of 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_CO_OP 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

Table 6-9 lists a new option 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_UNMARKED_BROWSE_MSG	To browse returning *only unmarked* messages.
MQGMO_MARK_BROWSE_HANDLE	To mark a message as browsed within the scope of this object 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 applications.
MQGMO_UNMARK_BROWSE_CO_OP	To unmark a previously browsed message for all cooperative applications.

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 messages from remaining unconsumed after they have been browsed and marked, for example, after a cooperative dispatcher fails after marking a message.

6.3.7 Examples

In this section we provide a description of a simple browser using browse and mark and of a cooperative dispatcher.

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 no 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 */
```

These options can be read as giving you the first (if any) message that you have not seen yet, then marking it so that you do not see it again.

If the browser application decides to consume the message or pass it to another application for processing, it can use the message token returned to identify the particular message for a later MQGET 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 */
```

These options can be read as giving you the first (if any) message not seen by any dispatcher, then marking it so that no dispatcher sees it again.

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 Callback for asynchronous consumers

Callback 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 callback functions passing messages to them. The callback 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 environment in which the program is running.

The first threading mode is that the queue manager steals the application program thread to do the callback dispatching and the application program waits (MQCTL with MQOP_START_WAIT) until all the callback 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 callback 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 callback consumer functions. The MQCTLO_THREAD_AFFINITY option in MQCTL specifies that all invocations of the consumer functions for the same connection will occur on the same thread. This requirement depends on the design of synchronous application event processing in a multi-threaded program.

6.4.2 Message consumers and event handlers

Callback 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 callback sequence describes the life cycle of the consumer:

1. The consumer function is registered by MQCB with the MQOP_REGISTER option.
2. The consumer is started by MQCTL with the MQOP_START option.
3. The consumer receives and processes messages.
4. Consumer is suspended and resumed by MQCTL with the MQOP_SUSPEND and MQOP_RESUME options or by MQCB with the MQOP_ SUSPEND and MQOP_RESUME options.
5. The consumer is stopped by MQCTL with MQOP_STOP option.
6. The 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 affects the entire callback environment for the connection, such as queue manager quiescing or connection stopping.

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

1. The event handler function is registered by MQCB with the MQOP_REGISTER option.
2. The event handler receives and processes events.
3. The event handler cannot be suspended, resumed, stopped, or deregistered. It ceases operation when the connection terminates.

6.4.3 MQCB manage callback

The MQCB function call registers a callback function for a specified object handle (queue or topic). A callback 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

The MQCB function syntax is:

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

Parameters

Table 6-11 lists the MQCB parameters.

Table 6-11 MQCB parameters

Parameter name	Data type	Input/output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the queue manager.
Operation	MQLONG	Input	Describes the operation being processed by MQCB. The supported operations are: ▶ MQOP_REGISTER ▶ MQOP_DEREGISTER ▶ MQOP_SUSPEND ▶ MQOP_RESUME
CallbackDesc	MQCBD	Input	Describes the callback 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 Callback Descriptor

This data structure identifies the callback function that is being registered and the options used during the registration. See Table 6-12.

Table 6-12 MQCBD Callback Descriptor

Parameter name	Data type	Input/output	Description
StrucID	MQCHAR4	Input	Structure identifier.
Version	MQLONG	Input	Current version is 1.
CallbackType	MQLONG	Input	Callback function types. There are two types: ▶ MQCBCT_MESSAGE_CONSUMER ▶ MQCBCT_EVENT_HANDLER
Options	MQLONG	Input	Options to indicate whether callback function is called to indicate the following message consumer state changes: ▶ MQCBDO_FAIL_IF_QUIESCING ▶ MQCBDO_REGISTER_CALL ▶ MQCBDO_START_CALL ▶ MQCBDO_STOP_CALL ▶ MQCBDO_DEREGISTER_CALL
MaxMsgLength	MQLONG	Input	Length in bytes of the longest message that can be read from the object handle and given to the callback routine. There is a value to indicate that messages should be delivered without truncation: MQCBD_FULL_MSG_LENGTH
CallbackFunction	MQCB_FUNC TION	Input	Pointer to the callback 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, specified as either CallbackFunction or CallbackName.
Reserved			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 callbacks

This function performs controlling actions on the callback functions. It executes the operations START, STOP, SUSPEND, and RESUME for message consumers.

There is also an operation called START with WAIT that suspends the calling application program until all the callback functions have been suspended, deregistered, or stopped. The callback functions run on same thread as the caller.

Syntax

The MQCTL function syntax is:

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

Parameters

Table 6-13 lists the MQCTL function parameters.

Table 6-13 MQCTL parameters

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

Example

The sample code in Example 6-8 shows how to register a message consumer callback function to process messages from a queue. This program starts the callback processing and then waits for input from the keyboard to stop callback processing and close the object and connection handles.

Example 6-8 MQCB and MQCTL C language example

```
/*   Declare MQI structures needed, use defaults where possible */
   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            */

   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 queuename <qmgrname> <getoptions>\n");
     exit(999);
   }

   /* Create object descriptor for subject queue */
   strncpy(od.ObjectName, argv[1], MQ_Q_NAME_LENGTH);
   QMName[0] = '\0';   /* default */
   if (argc > 2)
     strncpy(QMName, argv[2], MQ_Q_MGR_NAME_LENGTH);

   /* Connect to queue manager */
   MQCONN(QMName,                   /* queue manager               */
          &Hcon,                    /* connection handle           */
          &CompCode,                /* completion code             */
          &CReason);                /* reason code                 */
   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 MQ 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 callback function to be a message 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                  */
   if (Reason != MQRC_NONE)
      printf("MQCLOSE ended with reason code %d\n", Reason);
   }

   /* Disconnect from MQ (if connected) */
   if (Hcon != MQHC_UNUSABLE_HCONN)
   {
     if (CReason != MQRC_ALREADY_CONNECTED )
     {
       MQDISC(&Hcon,                  /* connection handle          */
              &CompCode,              /* completion code            */
              &Reason);               /* reason code                */
      if (Reason != MQRC_NONE)
         printf("MQDISC ended with reason code %d\n", Reason);
      }
   }
```

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

6.4.6 Callback 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 callback function is allowed to be registered per object handle using the same queue manager connection handle. If a second callback function is registered for the same object 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 callback functions for the same queue using different queue manager connection handles. In this case messages from the queue are processed in parallel and the sequence cannot be guaranteed.

Syntax

The callback function signature is:

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

Parameters

Table 6-14 shows the parameters received by the callback function.

Table 6-14 Callback function input parameters

Parameter name	Data type	Input/output	Description
Hconn	MQHCONN	Input	Handle that 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 Callback functions.

MQCBC callback context

This data structure is passed to callback functions as the context information. See Table 6-15.

Table 6-15 MQCBC callback 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 seven 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.

Field name	Data type	Input/output	Description
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 MQCBCF_BUFF_EMPTY flag may be returned when there has been an error during the read ahead process.
CallbackArea	MQPTR	Input/output	Field available for the callback function to use. This area is saved and returned unchanged by the queue manager. It can be used by the callback to store state information that is preserved across invocations of the function. This area is shared by callback function invocations with this object handle.
ConnectionArea	MQPTR	Input/output	Field available for the callback function to use. This area is saved and returned unchanged by the queue manager. It can be used by the callback to store state information that is preserved across invocations of the function. This area is shared for all callback functions that have the same connection handle.

Example

This callback function prints a message for each type of callback received. Refer to Example 6-9.

Example 6-9 Callback 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("Message Call : CallType = %d\n",pContext->CallType);
        printf("Message received\n");
        ... (process the message)
        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 Object Descriptor (MQOD) data structure has been extended to support the use of topics when the MQOPEN or MQPUT1 function 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 reference to a MQSD structure to register a subscription to receive publication messages.

The topic can be resolved 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 10,240 characters.
- 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

Subscriber applications are programs that receive messages previously sent to topic destinations. A subscriber application needs to do two things:

- Create one or more subscriptions, each of which identifies topic strings of interest and identifies a queue to place those messages on.
- Consume the messages delivered as a result of the subscriptions.

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

There are two types of subscriptions:

- Durable: These subscriptions remain active after the subscriber program closes the connection to the queue manager. All publications for the subscribed topic 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 or closes the connection to the queue manager.

There are two methods to identify where published messages are to be stored for delivery to a subscriber:

- Managed: These are destinations created by the queue manager and a handle is returned to the subscriber application. We recommend this for non-durable subscriptions, where messages may only be queued when the subscriber program is running.
- Non managed: These are destinations that the subscriber application supplies. They must be local queues that have already been opened. The handle is supplied to the subscription request. In this case, it is a responsibility of the subscriber 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 destination 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. Refer to Example 6-11 on page 132.

6.5.4 MQOD Object Descriptor

Table 6-16 lists the new fields added to 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 the ObjectName field 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. Used to specify a selection string to be used by MQGET calls.
ResObjectString	MQCHARV	Output	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

Table 6-17 lists the new Subscription Descriptor 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	Current version is 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 user ID 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.
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	Correlation ID set in the MQMD of all messages matching this subscription.
PubPriority	MQLONG	Input/output	Priority in the MQMD of all messages matching this subscription.

Field name	Data type	Input/output	Description
PubAccountingToken	MQBYTE32	Input/output	Accounting token the MQMD of all messages matching this subscription.
PubApplIdentityData	MQCHAR32	Input/output	Application identity data in the MQMD of all messages matching this subscription.
SelectionString	MQCHARV	Input/output	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	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

Table 6-18 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.
Version	MQLONG	Input	Current version is 3.
Options	MQLONG	Input	New options for publication: ▸ MQPMO_SUPPRESS_REPLYTO ▸ MQPMO_RETAIN
... other fields ...			Other existing fields precede the following.
ResolvedQName	MQCHAR48	Output	Returned with an undefined value for topics.
ResolvedQMgrName	MQCHAR48	Output	Returned with an undefined value for topics.

Field name	Data type	Input/output	Description
--- Other fields ...			Other fields that 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	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

Table 6-19 lists the changes to the MQ Message Descriptor (MQMD).

Table 6-19 MQMD changes

Field name	Data type	Input/output	Description
StrucId	MQCHAR4	Input	Structure Identifier.
Version	MQLONG	Input	Current version is 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 received the publication.
... other fields ...			Other existing fields precede the following.
Priority	MQLONG	Input/output	New value that can be specified is MQPRI_PRIORITY_AS_TOPIC_DEF.

Field name	Data type	Input/output	Description
... 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.
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

The MQSUB function syntax is:

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

Parameters

Table 6-20 lists the MQSUB parameters.

Table 6-20 MQSUB parameters

Parameter name	Data type	Input/output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the queue manager.
SubDesc	MQSD	Input/output	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	Subscription handle that 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.
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 wildcards then this request can generate multiple messages returned to the subscriber.

Syntax

The MQSUBRQ function syntax is:

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

Parameters

Table 6-21 lists the MQSUBRQ parameters.

Table 6-21 MQSUBRQ parameters

Parameter name	Data type	Input/output	Description
Hconn	MQHCONN	Input	Handle that represents a valid connection to the queue manager.
Hsub	MQHSUB	Input	Handle that represents the subscription.
Action	MQLONG	Input	Controls the action required with the subscription: MQSR_ACTION_PUBLICATION. This requests 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, topic, or subscription. The MQCLOSE options may specify whether durable subscriptions are kept or removed.

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

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

Syntax

The MQCLOSE function syntax is:

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

Parameters

Table 6-22 lists the MQCLOSE parameters.

Table 6-22 MQCLOSE parameters

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

Examples

Example 6-10 and Example 6-11 on page 132 are samples 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);
```

Example 6-11 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 and 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 put to the queue.
- MQACTP_FORWARD: This is a previously retrieved message that may have been modified is 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 must 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 filter 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.

For further details and examples message selectors refer to 4.3, “Selectors” on page 49.

6.8 Other MQI considerations

Also consider the following information:

- 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 book. Refer to the WebSphere MQ V7.0 manuals *Using C++*, *Using Java*, and *Using .NET* for detailed information. These are available in the WebSphere MQ V7.0 Information Center at:

  `http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

7

WebSphere MQ Java Message Service enhancements

This chapter discusses enrichments made to the WebSphere MQ classes for Java Message Service (JMS). These enhancements enable WebSphere MQ to be a more natural fit to the features of JMS.

The following sections describe the enhancements:

7.1 Read ahead

The read ahead feature in WebSphere MQ V7.0 allows messages from a destination to be streamed to the WebSphere MQ classes for JMS ahead of the JMS application requesting the messages. This saves the classes from having to send a separate request to the WebSphere MQ queue manager for each message that the JMS application consumes and allows the messages to be consumed with improved performance.

A JMS application uses a MessageConsumer to receive messages from the destination. It can use one of the receive methods from the MessageConsumer to fetch messages synchronously from the destination. It can also register an object that implements the MessageListener interface to fetch messages asynchronously.

Note: MessageConsumer, MessageListener, and receive methods are JMS concepts. For more information about these topics refer to the *JMS Specification*, available at:

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

A JMS application consumes both persistent and non-persistent messages from the destination either synchronously or asynchronously. When a JMS application needs to consume non-persistent messages, the destination can be configured such that the WebSphere MQ classes for JMS make use of an internal buffer in memory to read ahead and store the messages of interest before delivering them to the application.

The read ahead feature is only applicable for non-persistent messages. If persistent messages were read into a buffer, the queue manager would no longer be able to recover the messages in the event of failure. Hence, this capability is not supported.

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 performance 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 must be consumed rapidly in the order they are queued.

Figure 7-1 shows a JMS application receiving messages from a WebSphere MQ V6.0 queue manager. The JMS application must issue a receive for every message that it must consume from the queue.

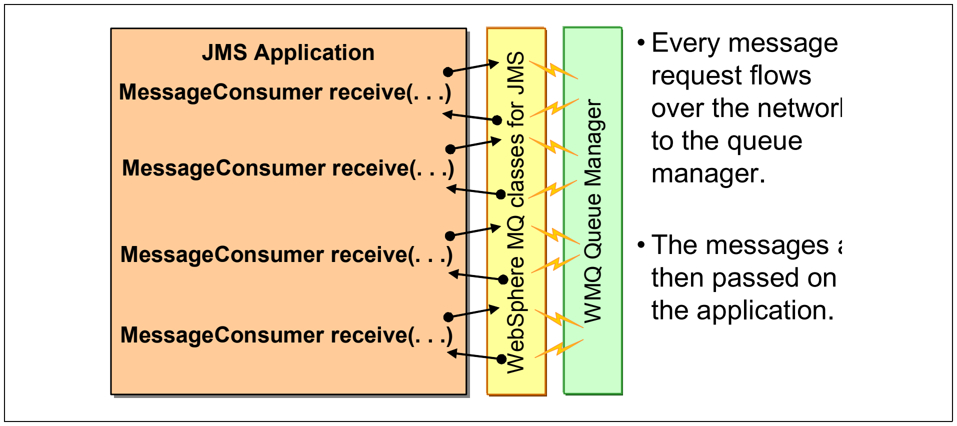

JMS Application
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
WebSphere MQ classes for JMS
WMQ Queue Manager
• Every message request flows over the networ to the queue manager.
• The messages then passed on the application.

Figure 7-1 JMS application getting message from WMQ V6.0 queue manager

Compare this with Figure 7-2, which shows a JMS application retrieving messages from a WebSphere MQ V7.0 queue manager using the read ahead feature.

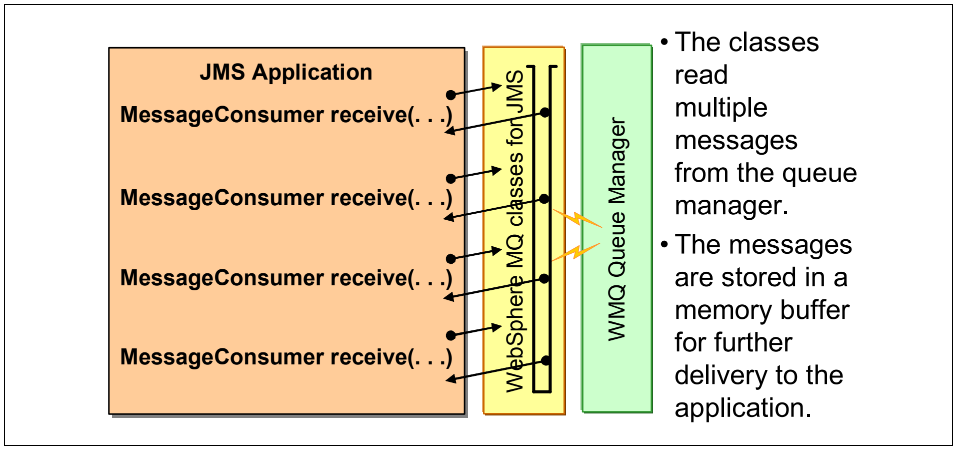

JMS Application
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
MessageConsumer receive(. . .)
WebSphere MQ classes for JMS
WMQ Queue Manager
• The classes read multiple messages from the queue manager.
• The messages are stored in a memory buffer for further delivery to the application.

Figure 7-2 JMS application getting messages from WMQ V7.0 queue manager using read ahead feature

The key considerations for using the read ahead feature in application designs are:

- If the JMS application using read ahead 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 the messages are lost.
- The read ahead feature is only applicable to non-persistent messages.
- Any existing JMS application can make use of the read ahead feature without any 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 or more message consumers in the same session to consume the messages from the queue.
  - Each message consumer uses a different destination object for the WebSphere MQ queue.
  - Any or both of the destination objects are configured for read ahead.

Note: The attribute READAHEADALLOWED must be set on the destination for the JMS application to make use of the read ahead feature in WebSphere MQ V7.0. Refer to 5.4, "Read ahead" on page 68.

7.2 Asynchronous put

Consider a Java Message Service (JMS) application that is responsible for collating and distributing weather samples at frequent intervals, for example, measurements of humidity, temperature, wind, rain, or snow. This application sends sequences of messages in rapid succession to the destination and does not require acknowledgement for every message sent.

Prior to WebSphere MQ V7.0, a JMS application connecting to a destination in client mode did not have very good performance when publishing messages in rapid succession. The WebSphere MQ classes for JMS had to wait for a response back from the queue manager for every message sent by the JMS application. Only after receiving the response from the queue manager would the WebSphere MQ classes for JMS return control back to the JMS application. The JMS application was then able to send the next message in the weather sample.

Using the enhanced asynchronous put feature in WebSphere MQ V7.0, the JMS application can now send messages in rapid succession with improved performance. The WebSphere MQ classes for JMS forwards each message to the queue manager and does not wait for a response. Control is immediately returned back to the JMS application and the application can proceed to send the next message.

Note: JMS applications can make use of the asynchronous put feature only when connecting to WebSphere MQ V7.0 in client mode. This feature is not applicable to binding mode.

In client mode, the application establishes connection with a WebSphere MQ queue manager using sockets over Transmission Control Protocol/Internet Protocol (TCP/IP).

In binding mode, the application establishes a connection with a WebSphere MQ queue manager using shared memory. Both the JMS application and the WebSphere MQ queue manager must be running on the same machine.

Existing JMS applications can make use of the WebSphere MQ V7.0 asynchronous put feature without any modifications to the code. For any JMS application to use this feature, the destination on the WebSphere MQ queue manager can be configured independently. For more information about configuring the destination for asynchronous put refer to 5.5, “Asynchronous put” on page 71.

WebSphere MQ classes for JMS function this way for persistent as well as non-persistent messages sent during a transacted session.

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

For messages sent in a transacted session, the JMS application determines whether the queue manager received the messages safely or otherwise, by committing the transaction.

For the non-persistent messages sent in a non-transacted session, the SENDCHECKCOUNT property of the ConnectionFactory object determines the number of messages to be sent, before WebSphere MQ classes for JMS checks the queue manager for acknowledged messages.

Note: For ConnectionFactory object attributes refer to 7.6, “Properties of WebSphere MQ classes for Java Message Service” on page 148.

A JMS application may encounter exceptions when sending messages in rapid succession. In such conditions, it is necessary for the JMS application to register an ExceptionListener with the connection object. In the event of failure, that is, when the WebSphere MQ server determines failure in message delivery, the WebSphere MQ classes 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 it accordingly.

Note: ExceptionListener is JMS terminology. For more information about ExceptionListener refer to the *JMS specification* available at:

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

Asynchronous put provides considerable performance benefits to JMS applications that transmit a sequence of messages in rapid succession but do not require immediate acknowledgement from the queue manager for every sent message.

Figure 7-3 shows the workflow of a JMS application sending messages in client mode to a WebSphere MQ V6.0 queue manager.

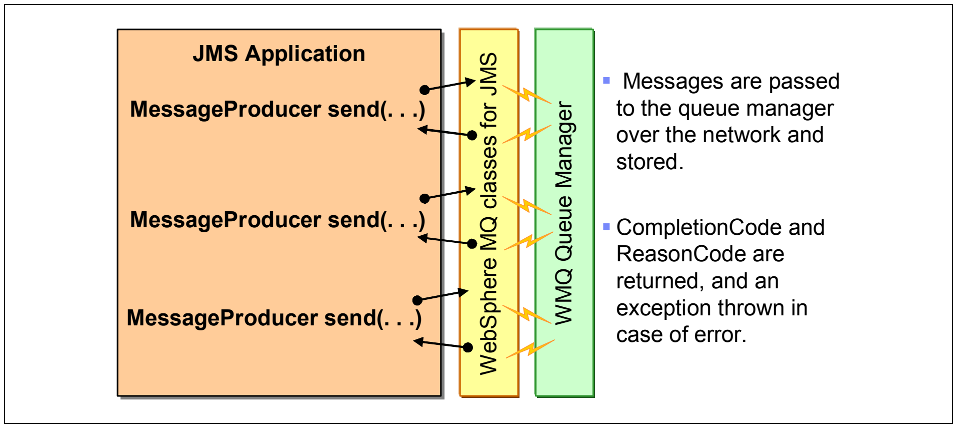

JMS Application
MessageProducer send(. . .)
MessageProducer send(. . .)
MessageProducer send(. . .)
WebSphere MQ classes for JMS
WMQ Queue Manager
Messages are passed to the queue manager over the network and stored.
CompletionCode and ReasonCode are returned, and an exception thrown in case of error.

Figure 7-3 JMS sending messages in client mode in WMQ 6.0

Compare this with Figure 7-4, which shows the workflow of a JMS application sending messages in client mode to a WebSphere MQ V7.0 queue manager using asynchronous put.

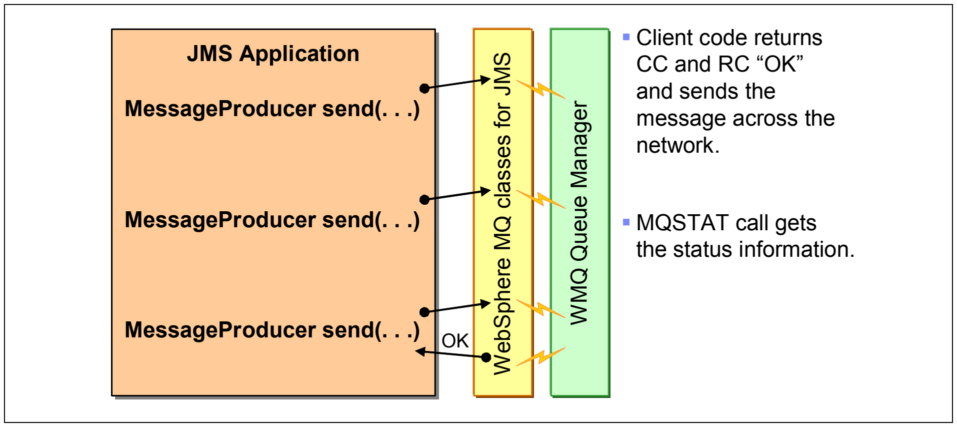

JMS Application
MessageProducer send(. . .)
MessageProducer send(. . .)
MessageProducer send(. . .)
OK
WebSphere MQ classes for JMS
WMQ Queue Manager
Client code returns CC and RC "OK" and sends the message across the network.
MQSTAT call gets the status information.

Figure 7-4 JMS sending messages in client mode with asynchronous put

7.3 Asynchronous consume

Prior to WebSphere MQ V7.0, asynchronous message consumption was not natively supported. To perform asynchronous message consumption, the WebSphere MQ classes for JMS periodically polled the destination for suitable messages to arrive.

Both synchronous and asynchronous message consumption are now supported as native features in WebSphere MQ V7.0. An application that needs to consume a message asynchronously registers a callback function for a destination. When a suitable message is available at the destination, WebSphere MQ calls the function and passes the message as a parameter. The function can then process the message asynchronously.

From the JMS application design or programming perspective there are no changes for asynchronous message consumption. As previously, the JMS application registers an object implementing the JMS MessageListener interface to consume messages asynchronously.

The implementation of JMS message listeners is now a natural fit with WebSphere MQ. In previous versions of WebSphere MQ, the WebSphere MQ classes for JMS polled the destination at a regular interval to check whether any suitable messages were available at the destination. Polling created additional workload at the application side because the WebSphere MQ classes for JMS had to send polling requests to the queue manager over the TCP/IP network and wait for a response. In the usual case where the destination was empty this resulted in inefficient usage of network resources and increased CPU usage on both the JMS application machine and the queue manager machine.

In WebSphere MQ V7.0, the WebSphere MQ classes for JMS no longer polls a destination to check the availability of a message. As soon as a suitable message arrives at the destination the WebSphere MQ classes for JMS pass the message to the MessageListener callback function.

The advantages of using WebSphere MQ native asynchronous consume are summarized as follows:

- Improved performance of JMS message listeners, particularly when an application uses multiple message listeners in a session to monitor multiple destinations.
- Reduced CPU usage at both the JMS application and the WebSphere MQ queue manager.
- Fewer requests over TCP/IP and a reduction in network traffic between the classes and the queue manager.
- Message throughput is increased and the time taken to deliver a message to a message listener is reduced.

Similar enrichments are seen when message-driven beans (MDBs) are used to retrieve messages asynchronously. Instead of the MDBs polling for messages, WebSphere MQ directly passes the messages to be consumed by the MDB. In some situations there can be multiple MDBs contending for messages from the same destination. The contention for messages results in higher CPU utilization. With the introduction of asynchronous consumption in WebSphere MQ V7.0, multiple MDBs that are consuming messages from the same destination now experience reduced message contention. This results in higher message processing performance.

7.4 Conversation sharing sessions

Conversation sharing is a new feature in WebSphere MQ V7.0. It allows a single TCP/IP socket to multiplex multiple connections, provided that the two ends of the connection belong to the same process. All Java Message Service applications by default use multiplexing of sessions without any code modifications.

In previous versions of WebSphere MQ, each instance of the JMS session created by the same parent JMS connection would use a different socket connection to connect to a queue manager. No two JMS sessions shared a 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 being established for every JMS session and more I/O resource utilization on both the application machine and the queue manager machine. This is depicted in Figure 7-5.

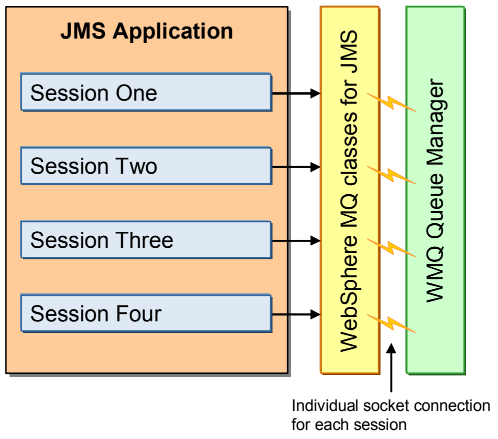


Figure 7-5 JMS multi-threaded clients using WMQ V6.0

With the introduction of conversation sharing in WebSphere MQ V7.0, multiple JMS sessions created from the same connection can be multiplexed across a single TCP/IP socket, as depicted in Figure 7-6 on page 146. This means that a JMS application creating multiple JMS sessions from a single connection object may now use a single TCP/IP socket. A consequence is that both ends of the socket now have threads that are always receiving data on the socket. This allows heartbeats to travel down the socket from both the ends. In the event of an I/O failure, the WebSphere MQ JMS classes can immediately identify the connection breaking. The JMS application can be more responsive in identifying the failure at all times and not just when an MQGET is outstanding. Conversation sharing thereby reduces the dependency of KEEPALIVE on TCP/IP to detect failures.

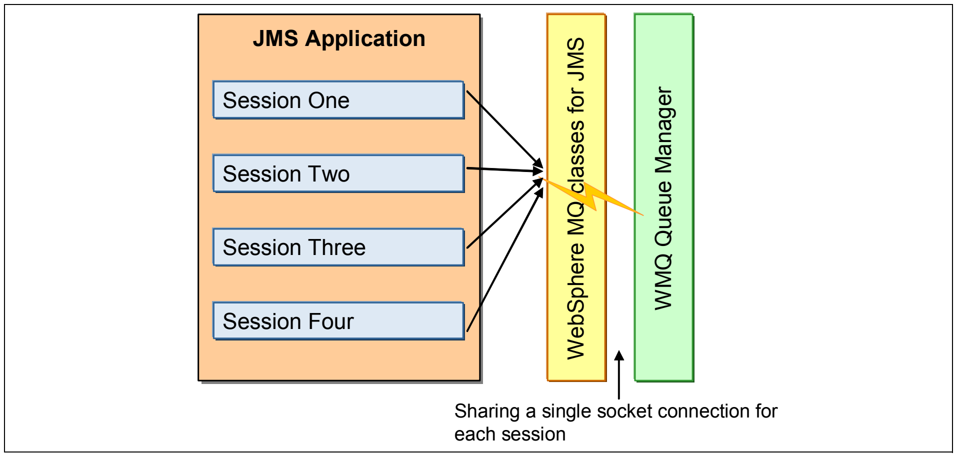

JMS Application
Session One
Session Two
Session Three
Session Four
WebSphere MQ classes for JMS
WMQ Queue Manager
Sharing a single socket connection for each session

Figure 7-6 JMS multi-threaded clients using WMQ V7.0

Note: JMS connection and JMS session are JMS terms. For more information about these topics refer to the *JMS Specification* available at:

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

KEEPALIVE is a TCP/IP property that specifies the duration for which the connection should be maintained if there is no activity on the socket.

For more information about conversation sharing refer to 5.3, “Conversation sharing” on page 64.

7.5 Selectors and mapping of MQ and JMS messages

Java Message Service applications can use message selectors for filtering messages from the destination. An application would thus receive those messages containing properties that match the selector string, as specified by the application. In WebSphere MQ V6.0, the queue manager did not natively support the message selection mechanism. The JMS application had to browse the queue and perform the message selection by examining each message. This resulted in high CPU usage on both the application machine and the queue manager machine.

A JMS message consists of a header that is made up of message properties and a body that contains the application data. As a minimum, a WebSphere MQ message consists of a MQ Message Descriptor and the message data, as shown in Figure 7-7.

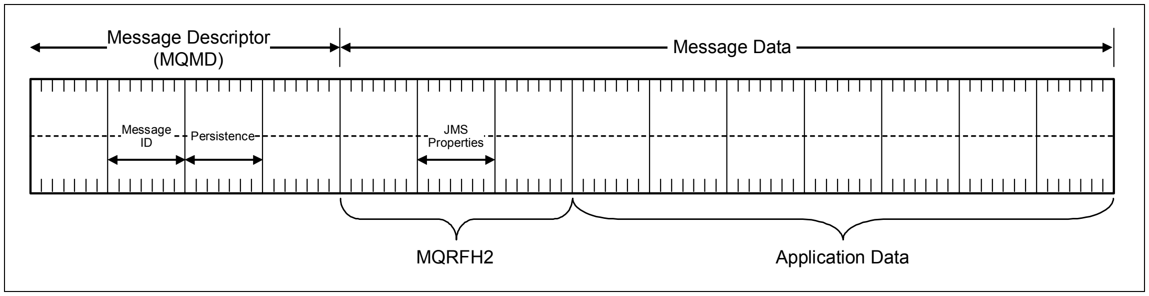


Figure 7-7 Representation of a MQ message

A JMS application can now read and write the full WebSphere MQ message including the MQ Message Descriptor. JMS applications can now provide:

- A custom message ID for a message
- A custom user ID for a message
- A custom application name that puts the message
- Finer control over other MQ Message Descriptor fields

A mechanism has been introduced to receive WebSphere MQ messages within JMS applications as the body of a JMS BytesMessage.

Other MQ API applications can now access the JMS user properties by using new MQI calls for handling message properties. They will now be able to:

- Read and write MQ Message Descriptor fields and JMS message properties that were set by JMS applications.
- Access the application data in messages produced by JMS without needing to parse the message property header.

WebSphere MQ V7.0 provides native support for message selectors. The selection of messages is performed by the queue manager. When a JMS application attempts to retrieve a message from the destination, the queue manager filters the messages that match the selector and delivers them to the application. Message properties also enable message selection by applications that use the Message Queue Interface (MQI).

Previously, problems could be encountered when inter-operating messages between JMS applications and MQI Applications. The JMS applications could not access the message properties in the MQ Message Descriptor (MQMD). The MQ API 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 while using JMS or MQ API with the same property name. For more information about message selectors, refer to 4.3, “Selectors” on page 49.

When a JMS application sends a message, the WebSphere MQ classes for JMS map the JMS message into a WebSphere MQ message. Some of the JMS header fields and properties are mapped into fields in the MQ Message Descriptor and others are mapped into fields in an additional WebSphere MQ header called an MQRFH2. On receipt of a JMS message by the JMS application, WebSphere MQ classes for JMS performs the reverse mapping. An application that is using the MQ API to receive messages from a JMS application must therefore be able to handle MQRFH2. If the application cannot handle MQRFH2, the TARGCLIENT property of the destination object can be set to tell WebSphere MQ classes for JMS not to include MQRFH2 in the WebSphere MQ messages. However, by excluding the MQRFH2, the information held in some of the JMS header fields and properties will be lost.

Similarly, an application that is using the MQ API to send messages to a JMS application must include an MQRFH2 in each message. If an MQRFH2 is not included, WebSphere MQ classes for JMS can set only those JMS header fields and properties that can be derived from the fields in a MQ Message Descriptor.

For more information about message properties refer to 6.2, “Message properties” on page 89.

7.6 Properties of WebSphere MQ classes for Java Message Service

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

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

Table 7-1 lists some of the new properties introduced in 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	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		
PROVIDERVERSION	PVER	The version, release, modification level, and fix pack of the queue manager to which the application intends to connect.	Y	Y	Y		
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		

Property	Short name	Description	CF/ XACF	QCF/ XAQCF	TCF/ XATCF	Q	T
SHARECONVALLOWED	SCA	Whether a client mode 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		
WILDCARDFORMAT	WCF MT	The version of wildcard syntax to be used.	Y		Y		Y

Besides these properties, WebSphere MQ V7.0 introduces new attributes that can be used with the JMS administrative objects. For a complete reference of the properties and their detailed description see to the section “WebSphere MQ classes for JMS → Properties of objects” in the *Using Java* manual, available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

7.7 Tracing programs

The WebSphere MQ classes for JMS trace facility has been greatly enhanced in WebSphere MQ V7.0 to improve the ability of IBM to diagnose customer problems. Various properties are provided to control tracing. Trace is turned OFF by default.

Tracing is turned on by setting com.ibm.msg.client.commonservices.trace.status to ON. To turn tracing off, set this property to OFF.

Table 7-2 contains details of all the trace properties.

Table 7-2 Properties to configure the trace facility.

Trace property	Function
com.ibm.msg.client.commonservices.trace.outputName	The directory and file name to which trace output will be sent.
com.ibm.msg.client.commonservices.trace.include	A semicolon (;) separated list of packages and classes that will be traced, or the special values ALL or NONE.
com.ibm.msg.client.commonservices.trace.exclude	A semicolon-separated list of packages and classes that will not be traced, or the special values ALL or NONE.
com.ibm.msg.client.commonservices.trace.maxBytes	The maximum number of bytes that will be traced from any byte arrays.
com.ibm.msg.client.commonservices.trace.limit	The maximum number of bytes to be written to a trace output file.
com.ibm.msg.client.commonservices.trace.count	The number of trace output files to cycle through.
com.ibm.msg.client.commonservices.trace.parameter	Whether method parameters and return values are included in the trace.
com.ibm.msg.client.commonservices.trace.startup	Whether to trace the initialization phase of WebSphere MQ classes for JMS during which resources are allocated, including the initialization of the main trace facility.
com.ibm.msg.client.commonservices.trace.compress	Whether trace output is compressed.
com.ibm.msg.client.commonservices.trace.level	A filtering level for the trace.

For a complete reference of the properties and their detailed descriptions refer to the section "WebSphere MQ classes for JMS → Using WebSphere MQ classes for JMS → Solving problems" in the *Using Java* manual, available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

Note: The trace parameters used by versions of WebSphere MQ classes for JMS earlier than Version 7.0 are still supported. However, these should be considered deprecated for new applications.

Note: When trace is activated, it creates a file named mqjms.trc.lck. If the version of Java is earlier than Java 5, this file is not removed when the trace ends.

Administration enhancements

This chapter describes all the enhancements and changes to WebSphere MQ V7.0 for administration functions. It discusses WebSphere MQ Explorer, control commands, and MQSC administration interfaces.

This chapter contains the following sections:

Publish/Subscribe administration is not covered in this chapter due to the complexity of this new feature. Refer to Chapter 9, “Publish/Subscribe management” on page 197, for details of Pub/Sub administration tasks.

8.1 WebSphere MQ Explorer

The Eclipse-based graphical tooling, MQ Explorer, introduced in the previous release, has been extensively enhanced 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 and can be installed on machines without charge.

This section describes what is new in MQ Explorer. The main enhancements are:

- General GUI enhancements: MQ Explorer has a new welcome page and many linked pages to assist inexperienced users. It is now possible to export and import MQ Explorer settings.
- Browsing messages: The window has been enhanced to display message properties. Message properties are a new extension to messages in WebSphere MQ V7.0.
- Mapping between MQ objects and JMS objects: New JMS-administered objects can be created based on existing MQ objects, or new MQ objects can be created based on existing JMS objects.
- Remote administration of local queue managers: A new window has been added that can check and enable the prerequisites to allow for remote administration of local queue managers.
- Security: Improved capabilities to secure administration connections to queue managers.
- Queue manager sets: Queue managers can be grouped into the sets to perform administration tasks instead of performing the task 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, so MQ administrators can profit from these MQ Explorer enhancements to support their V6 environments.

8.1.1 General GUI enhancements

Tip: These enhancements can also be used for V6 queue managers.

There are two general enhancements to MQ Explorer GUI. The welcome page brings a better start point for using MQ Explorer by new or less experienced users. The ability to export and import MQ Explorer settings is useful for product reinstallation or to transfer settings to other instances of MQ Explorer.

Welcome page

This page automatically displays when MQ Explorer is launched for the first time.

Tip: To display the welcome page at any time, select **Help** → **Welcome** from the menu bar.

Figure 8-1 shows the new MQ Explorer welcome page.

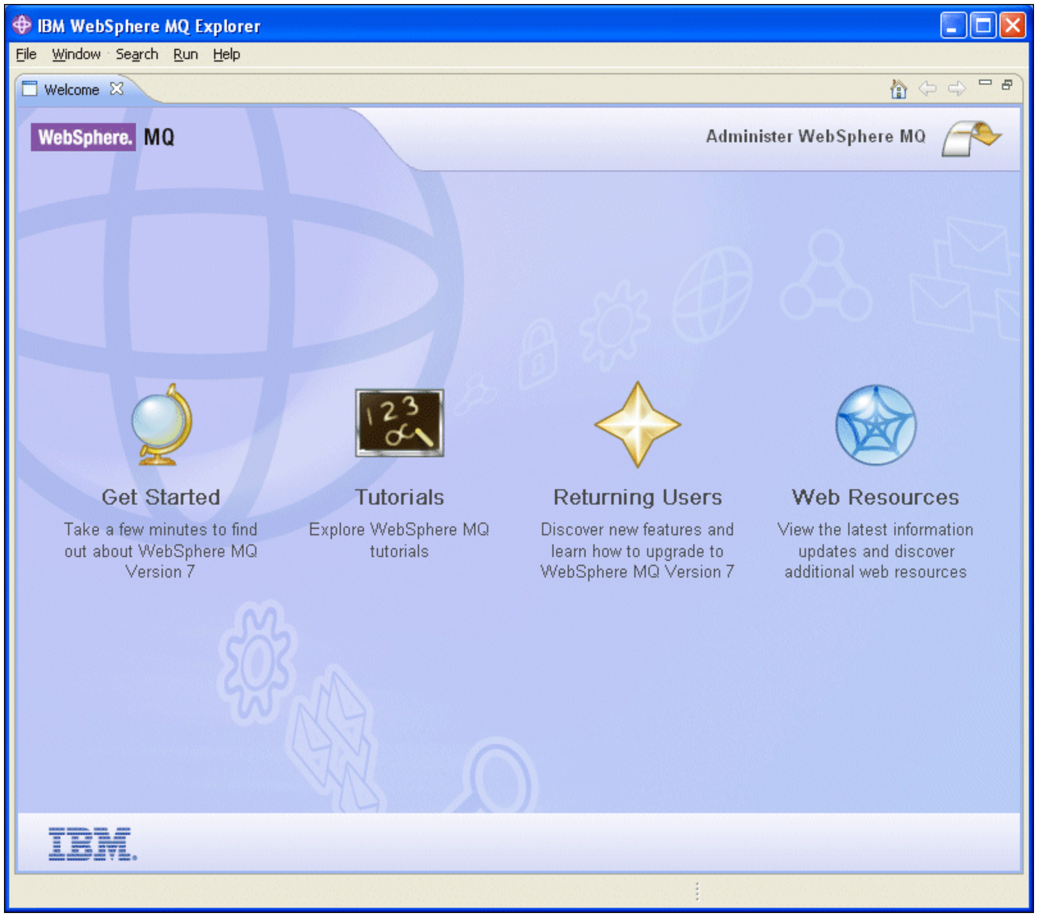

IBM WebSphere MQ Explorer
File Window Search Run Help
Welcome
WebSphere. MQ
Administer WebSphere MQ
Get Started
Take a few minutes to find out about WebSphere MQ Version 7
Tutorials
Explore WebSphere MQ tutorials
Returning Users
Discover new features and learn how to upgrade to WebSphere MQ Version 7
Web Resources
View the latest information updates and discover additional web resources
IBM.

Figure 8-1 MQ Explorer welcome page

The Administer WebSphere MQ view allows navigation of the WebSphere MQ configuration to display and administer queue managers, clusters, and JMS objects. This is the main view of MQ Explorer for experienced users and administrators.

There are also four icons on the welcome page. From left to right they are:

- Get Started: This view contains links to learn about the basic features of the WebSphere MQ product, set up a default configuration for the demonstration applications, run the Postcard application, run the API exerciser application, and install the product documentation on to the local system. The documentation is also available on the Internet as PDF files in the WebSphere MQ Library at:

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

  The documentation is also available as an interactive tree view in the IBM Information Center at:

  `http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

- Tutorials: This view provides links to tutorials to learn how to perform basic tasks with WebSphere MQ such as creating a queue manager, creating a queue, creating a channel, putting a message onto a queue, and getting a message from a queue.
- Returning Users: This view is for users who are familiar with WebSphere MQ and wish to discover the new features in WebSphere MQ V7.0 and learn how to upgrade to the new version.
- Web Resources: This view contains links to various resources on the Internet about the WebSphere MQ product family, including education, technical resources and best practices, product extensions, IBM business partners, WebSphere MQ performance, related products, getting involved, and support.

Export and import settings

Settings can be exported from MQ Explorer to a file or imported from a file into MQ Explorer. This can be useful for backup purposes, product reinstallation, to transfer settings to another instance of MQ Explorer, or to cater for multiple installations of MQ Explorer.

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 **IBM WebSphere MQ** in the Navigator view, as illustrated in Figure 8-2.

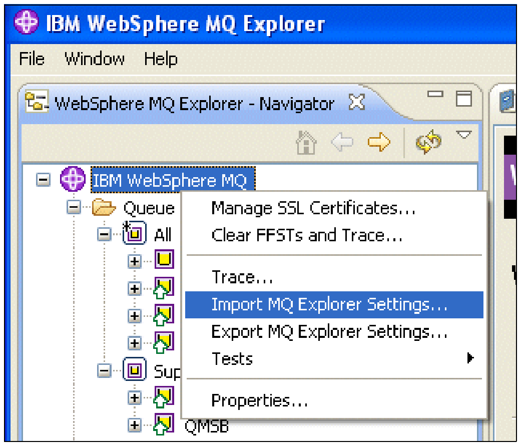
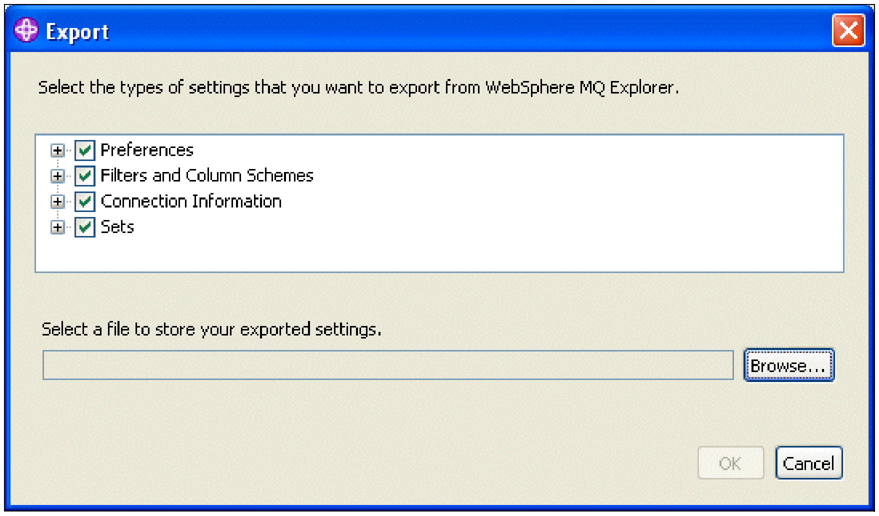

IBM WebSphere MQ Explorer
File Window Help
WebSphere MQ Explorer - Navigator
IBM WebSphere MQ
Queue
All
Sup
QMSB
Manage SSL Certificates...
Clear FFSTs and Trace...
Trace...
Import MQ Explorer Settings...
Export MQ Explorer Settings...
Tests
Properties...

Figure 8-2 Export and import settings in MQ Explorer V7.0

Exporting settings

To export settings, click **Export MQ Explorer Settings** and then select the settings to be exported and the location of the file to contain the settings. See Figure 8-3.


Export

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.

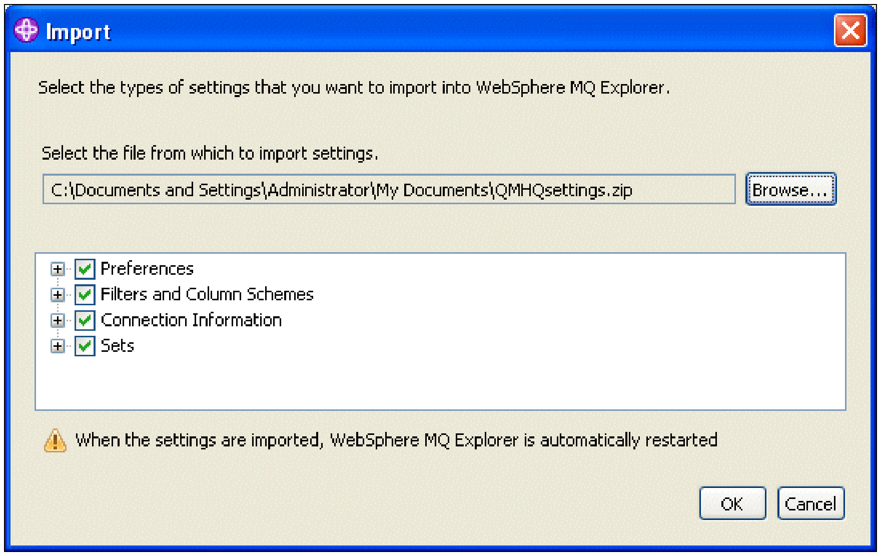
Figure 8-3 Export settings window

The exported settings are stored as XML in compressed ZIP file format.

Note: When exporting the 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 **Import MQ Explorer Settings** and then select the file for import and the settings to be imported. See Figure 8-4.



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 window

Note: For detailed information about how import works if the ZIP file contains manually created queue manager sets and filters, refer to the topic Configuring WebSphere MQ Explorer → Exporting and importing settings in the MQ Explorer Help menu.

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 windows, for example, queue status and topic status.

Note: For detailed information about V6 import restrictions, refer to the topic Configuring WebSphere MQ Explorer → Exporting and importing settings in the MQ Explorer Help menu.

8.1.2 Browsing messages

MQ Explorer can now display message properties. Message properties are a new extension to the structure of a WebSphere MQ message. There are also new preferences related to message browsing and the display of message properties.

New MQ Explorer preferences

Tip: This enhancement can also be used for V6 queue managers.

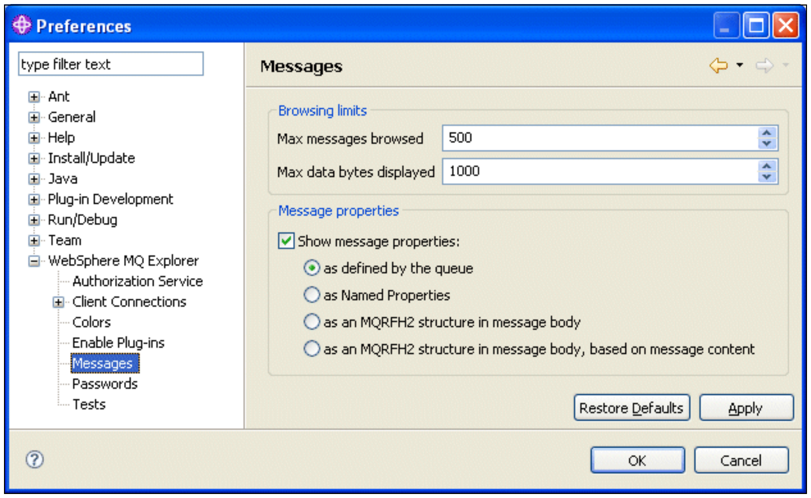
It is now possible to specify how many message can be browsed and what size of the message data is displayed:

- Max messages browsed: Maximum number of messages displayed in the Message browser window. The default value is 500.
- Max data bytes displayed: The number of bytes from the beginning of the message that are displayed. The default value is 1000.

The presentation format for displaying message properties of browsed messages can also be specified:

- Do not show message properties.
- Show message properties as defined by the queue being browsed, as per the property control (PROPCTL) attribute.
- Show message properties as name properties with values that are separate from the display of message data.
- Show message properties as name properties with values, and also show the MQRFH2 structure containing the properties in the display of message data.
- Only show message properties as name properties with values and show the MQRFH2 structure based on the message content.

To set these preferences go to the main menu and select **Window** → **Preferences**. In the left pane of the window displayed, select **WebSphere MQ Explorer** → **Messages**. See Figure 8-5.



Messages

Browsing limits

Max messages browsed: 500

Max data bytes displayed: 1000

Message properties

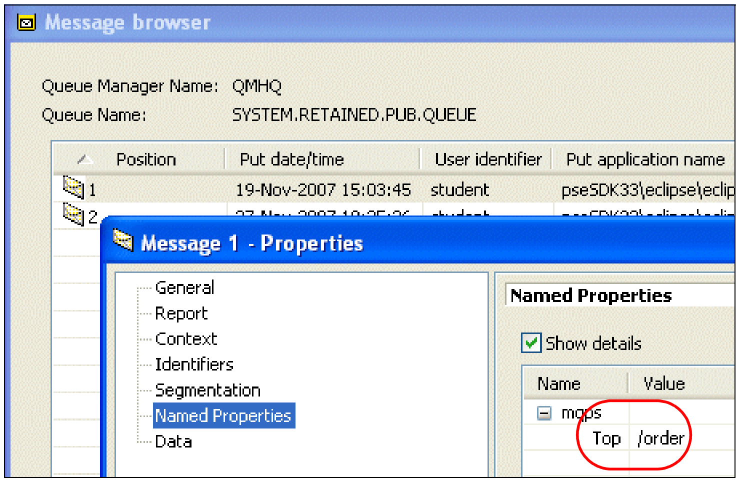
[✓] Show message properties:

- [● as defined by the queue]
- ○ as Named Properties
- ○ as an MQRFH2 structure in message body
- ○ as an MQRFH2 structure in message body, based on message content

Figure 8-5 Message browsing preferences

Browsing message properties

The new option *named properties* has been added to the message browsing window, as illustrated in Figure 8-6.



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

Name	Value
mqps	
Top	/order

Figure 8-6 Browsing message properties

This example shows a property in a Publish/Subscribe message. The property name *top* has the value /order.

8.1.3 Mapping between MQ objects and JMS objects

Tip: These enhancements can also be used for V6 queue managers.

These new MQ Explorer functions can be used to simplify administration tasks when creating MQ objects and JMS objects.

Creating a MQ object from a JMS object

New MQ queues and topics can be created based on existing JMS queues and topics. The values of relevant properties of the JMS object are copied to the new MQ object.

Important: If changes are made to a JMS object after the creation of the MQ object has been done, the changes are not reflected in the MQ object.

To create a new MQ object from an existing JMS object:

1. Expand the **JMS Administered Objects** tree in the Navigator view and select a 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.

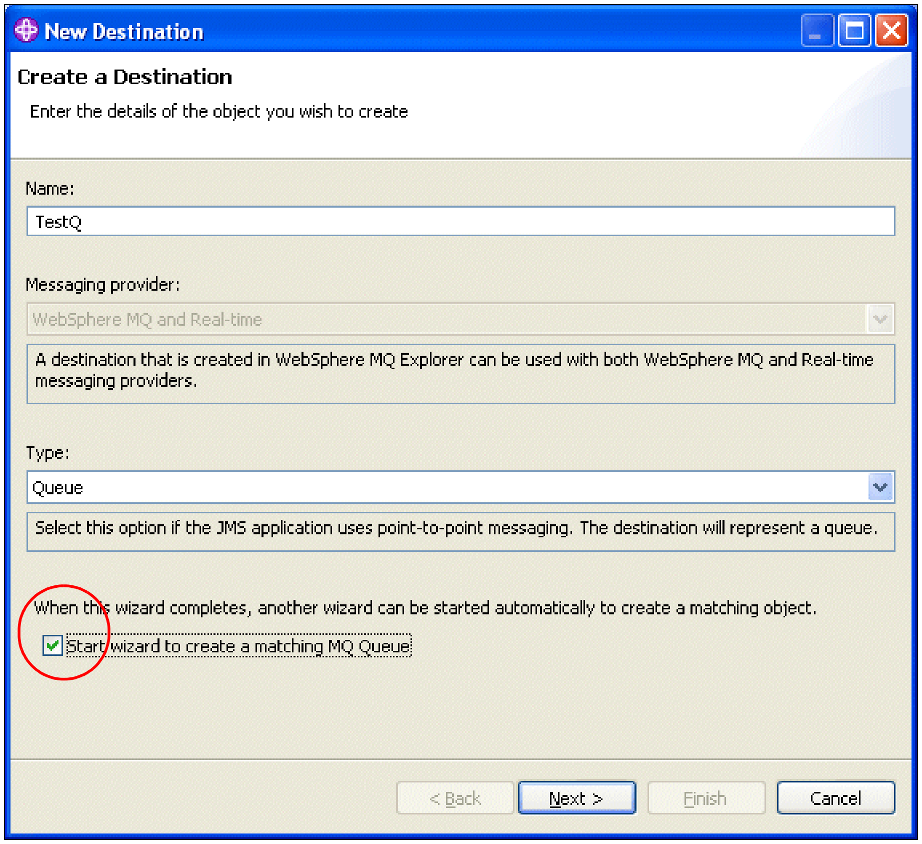
4. The new MQ object is created and displayed under the appropriate queue manager in MQ Explorer.

Creating a JMS object and an MQ object simultaneously

When creating a new JMS object (queue or topic) in MQ Explorer and 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 the **Start wizard to create a matching MQ** object check box, as illustrated in Figure 8-7, for the queue.


New Destination

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 is complete.

Note: Details on creating a JMS object from a MQ object are discussed in the following two topics in the MQ Explorer Help menu:

- Configuring JMS administered objects → Creating a JMS object from a WebSphere MQ object
- Configuring JMS administered objects → Creating a JMS object and an MQ object simultaneously

Creating a JMS object from a MQ object

New JMS queues and topics can be created based on existing MQ queues and topics. The values of relevant properties of the MQ object are copied to the new JMS object.

Important: If changes are made to a MQ object after the creation of the JMS object has been done, the changes are not reflected in the JMS object.

To create new JMS object from an existing MQ object:

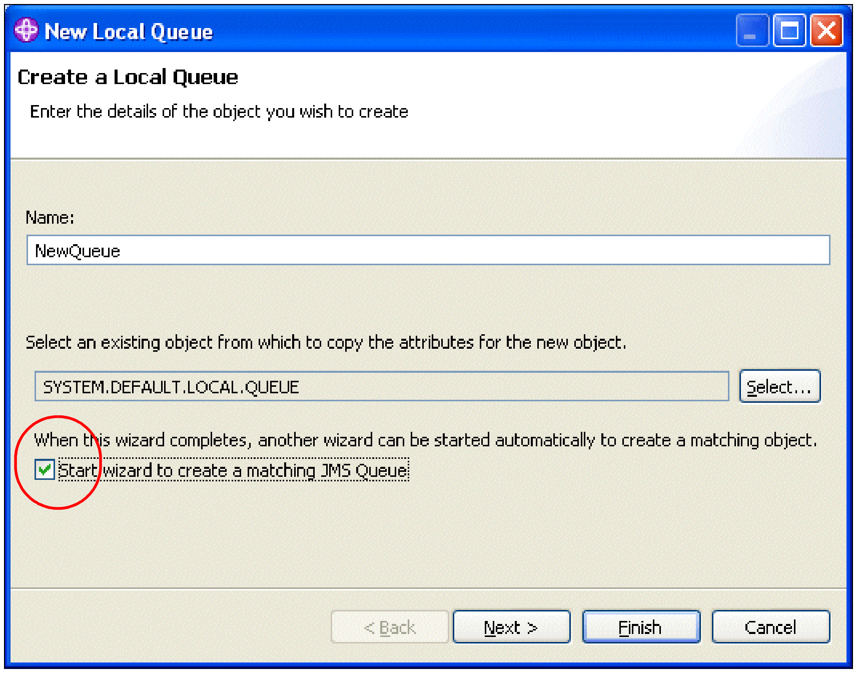
1. Expand the queue manager objects tree in the Navigator view and select source MQ object, queue, or topic in the Content view.
2. Right-click the object and select **Create JMS Queue** or **Create JMS Topic**. The wizard opens as appropriate.
3. Work through the wizard to define the new JMS object, 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 MQ object (queue or topic) in MQ Explorer and it has completed successfully, an appropriate JMS object can be created immediately.

To create a new MQ object and JMS object simultaneously:

1. Create a new MQ object as usual.
2. Select the **Start wizard to create a matching JMS object** check box, as illustrated Figure 8-8, in for the queue.



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 creating the new MQ object has finished.

Note: Details on creating a JMS object from a MQ object are covered in the following two topics in the MQ Explorer Help menu:

- Configuring JMS administered objects → Creating a JMS object from a WebSphere MQ object
- Configuring JMS administered objects → Creating a JMS object and an MQ object simultaneously

8.1.4 Remote queue managers administration

This new MQ Explorer feature can easily check and enable three of the prerequisites to allow remote instances of MQ Explorer to administer local queue managers.

Remote administration prerequisites window

This window simplifies the administration tasks for configuring remote administration.

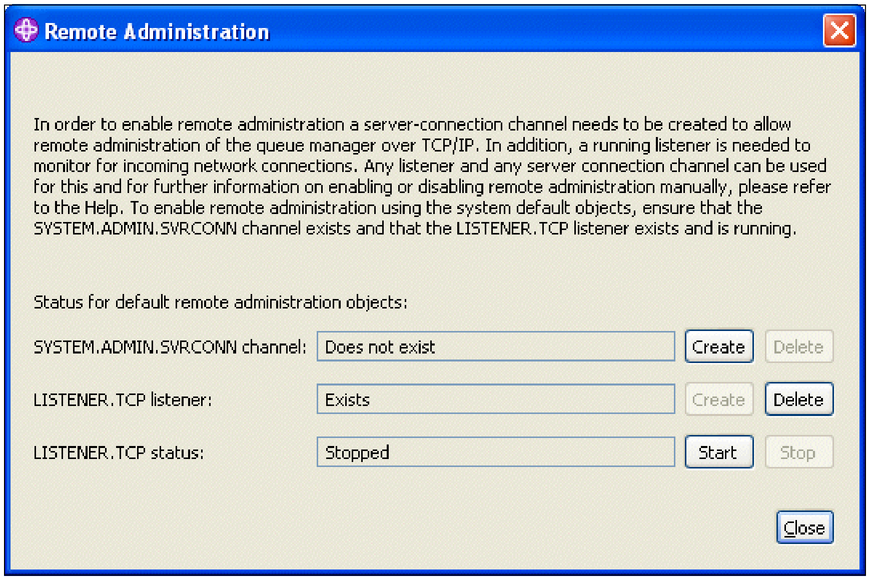
The following objects and settings are required for remote queue manager administration to be possible:

- The MQ command server must be running.
- A TCP/IP MQ listener must be running.
- The server-connection “SYSTEM.ADMIN.SVRCONN” must exist.
- The model queue “SYSTEM.MQEXPLORER.REPLY.MODEL” must exist.

The new Remote Administration window is available for local queue managers to check and create the following prerequisites:

- The server-connection channel named “SYSTEM.ADMIN.SVRCONN’ exists.
- A TCP/IP listener named “LISTENER.TCP” exists.
- The listener named “LISTENER.TCP” is started.

To use the feature, select the appropriate local queue manager, right-click it, then select **Remote Administration**. The Remote Administration window is displayed, as shown in Figure 8-9.



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 window

Notes: The Remote Administration option is only available if a queue manager is running.

The Remote Administration window does not included the MQ command server and the SYSTEM.MQEXPLORER.REPLY.MODEL model queue because these object are created by default when a queue manager is created.

This window can be used to create and delete the SYSTEM.ADMIN.SVRCONN channel and TCP/IP Listener by clicking the appropriate radio buttons. All actions are simple one-click operations, except Create for TCP/IP Listener, which requires input information about the TCP/IP port number. The window shows a warning if the port is already used by another queue manager.

Note: For detailed information refer to the topic Configuring WebSphere MQ → Administering remote queue managers → Enabling remote administration of queue managers in the MQ Explorer Help menu.

There are also security-related enhancements for remote queue managers administration that are covered in 8.1.5, “Security” on page 167.

Showing and hiding a remote queue manager

Tip: This enhancement can also be used for V6 queue managers.

There are some changes when working with remote queue managers:

- A remote queue manager can be directly added for administration instead of opening the Show/Hide Queue Managers window and going through its windows.

  To add a remote queue manager, right-click the **Queue Managers** folder in the Navigator view, then select **Add Remote Queue Manager**.
- A remote queue manager can be hidden without going through the Show/Hide Queue Managers window.

  To hide a remote queue manager, right-click the relevant queue manager name folder in the Navigator view, then select **Hide**.

It is still possible to add or hide queue managers using the Show/Hide Queue Managers window.

8.1.5 Security

Tip: These enhancements can also be used for V6 queue managers.

This section describes security-related enhancements for WebSphere MQ Explorer V7.0 that enable the use of more secure administration connections to queue managers.

The new enhancements help facilitate secure administration connection to a queue manager with a user ID and password, and to allow different security exits to be specified for each queue manager.

Note: It was only 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 in the following two places in MQ Explorer:

- In MQ Explorer preferences, under Client Connections, to be used as the default for new remote queue manager connections.
- For each remote queue manager connection.

The details about security exits for queue manager client connection are provided in 5.10, “Security exit details in WebSphere MQ Explorer” on page 82.

Note: The user ID and password are only authenticated by the remote queue manager if a suitable security exit is enabled in the server-connection channel on the remote queue manager.

This section only describes how security exits can be set up in MQ Explorer for remote queue manager administration purposes.

Default security preferences

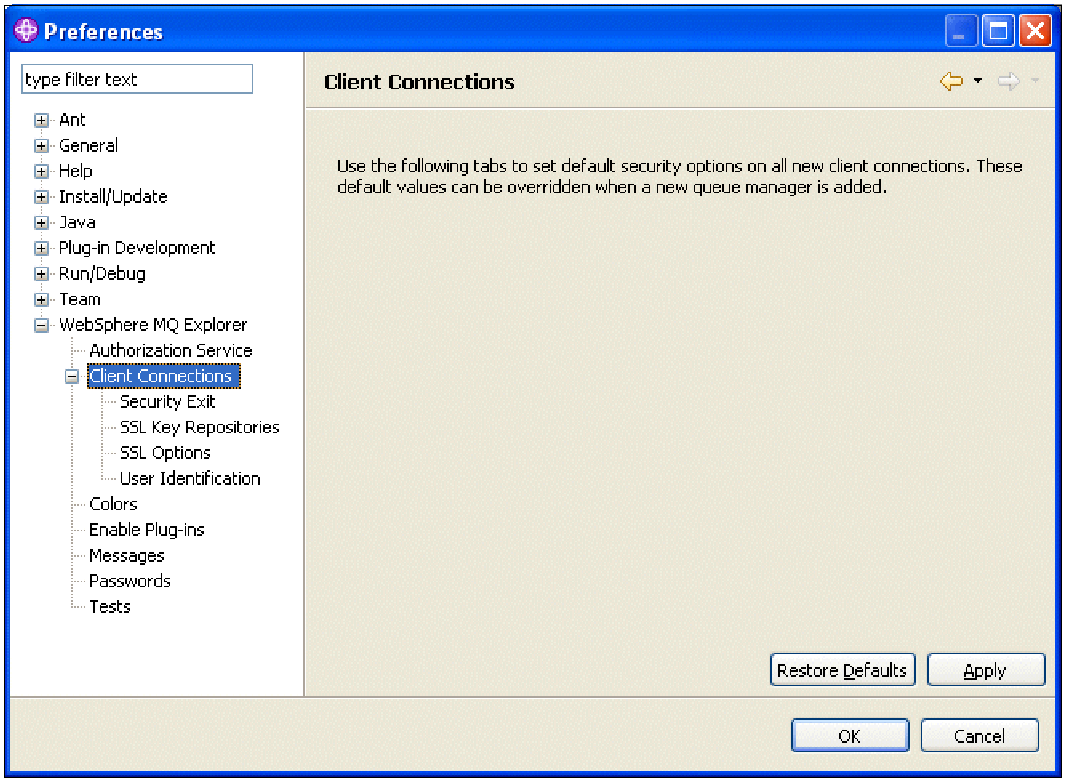
New options in MQ Explorer preferences have been added to support new security features:

- Client connections
- Passwords

Client connections

The default security exit and default user ID and password parameters for all the administered queue managers can be set and changed here. The parameters can be overridden by defining new security options when adding 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 window shown in Figure 8-10.



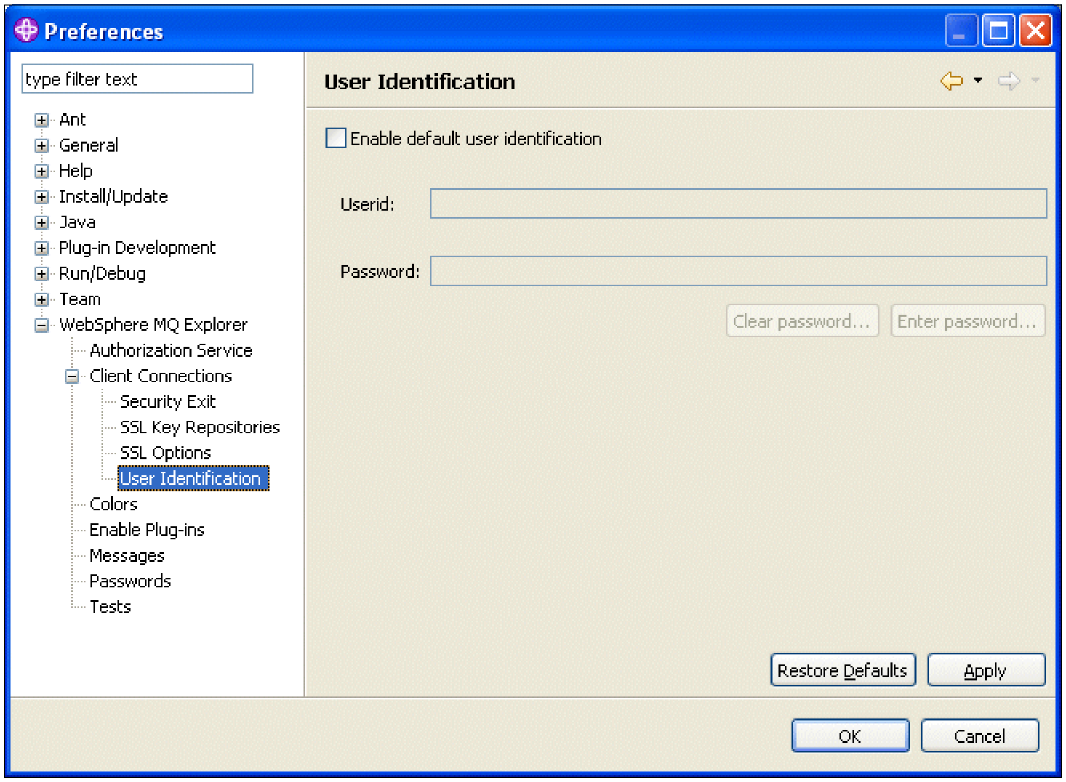
Use the following tabs to set default security options on all new client connections. These default values can be overridden when a new queue manager is added.

Figure 8-10 Default Client Connections settings

There are four sub-options under the Client Connections option:

- Security Exit: To set a default security exit name, path, and security exit data.
- SSL Key Repositories: To set a default SSL trusted certificate store and personal certificate store.
- SSL Options: To set a 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. It is now possible to secure administration connections to a queue manager with a user ID and password. This window is shown in Figure 8-11.



☐ Enable default user identification

Userid:

Password:

Figure 8-11 Default User Identification settings

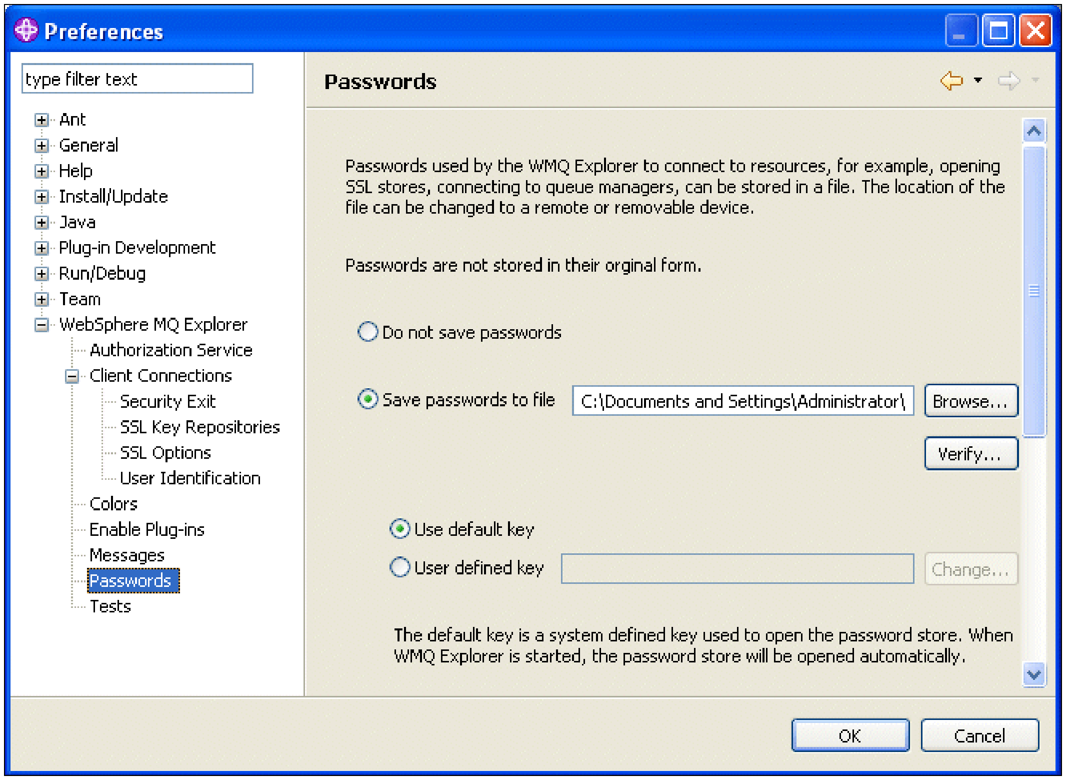
It is possible to only enter the user ID. This forces a password window to be displayed each time a connection is made. Alternatively, the password can be entered and saved. The password is stored in password store in a secure way. In this case, the parameters for password store must be set. This is described in the next section.

Note: For detailed information refer to the topic Configuring WebSphere MQ > Managing security and authorities → Configuring a default security exit > Default security preferences in the MQ Explorer Help menu.

Passwords

Passwords are used by the MQ Explorer to connect to resources, for example, opening SSL stores or connecting to queue managers. These passwords are securely 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 window shown in Figure 8-12.



Passwords

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

Enable or disable the password saving feature using the radio buttons. To save passwords to a file, choose the file to be used. To check whether the password store file exists and has appropriate permissions, click **Verify**.

With the Use default key option the password store opens automatically when it is needed. The the User defined key option means accesses to the password store require a specific password. In this case the password window 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 Configuring WebSphere MQ > Managing security and authorities → Configuring a default security exit > Passwords preferences in the MQ Explorer Help menu.

Security settings for remote queue manager connection

The Add Remote Queue Manager wizard has four new windows to define specific connection security details for each queue manager.

The first two windows are the same as for the previous version of MQ Explorer, which allowed basic connection parameters to be defined, such as queue manager name, TCP/IP address and port, and refresh interval. The wizard then navigates through the following new windows for the queue manager:

- Specify security exit details: To set default security exit name, path, and security exit data.
- Specify user identification details: To set the default user ID and password for client connections.
- Specify SSL certificate key repository details: To set default SSL trusted certificate store and personal certificate store.
- Specify SSL option details: To set 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 Configuring WebSphere MQ → Showing or hiding a queue manager > Showing a remote queue manager in the MQ Explorer Help menu.

8.1.6 Queue manager sets

Tip: This enhancement can be also used for V6 queue managers.

MQ Explorer now supports grouping of queue managers for the purpose of common administration. A group of queue managers is called a set. An action can be performed on a set of queue managers instead of performing the action on each queue manager one by one.

Membership of queue managers in a set can be based on many criteria, for example, the environment (test, production, and so on), the version of

WebSphere MQ (Version 6, Version 7), departments of the company, on the operating system, or a combination of specific values of many attributes. Queue managers may be members of none, one, or many sets.

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 automatically becomes a member of this set.

Grouping can be done manually simply by selecting the queue managers to be added to the set or automatically by filtering on:

- Command level
- Platform
- Any other queue manager attribute via a custom-made filter

Sets may not contain other sets. There is a default set called *all* that cannot be edited or deleted.

Working with the queue manager sets

The display of sets in MQ Explorer must be enabled. Although queue manager sets still exist when display is disabled, they cannot be managed.

Enable and disable displaying sets

To display sets:

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* that contains all queue managers.

To hide all sets:

1. Right-click the **Queue Managers** folder in the Navigator view.
2. Select **Sets** → **Hide Sets**.
3. The command hides all the defined sets, including the all set, from the Navigator view. The sets are not deleted, they are just hidden from view.

The Sets option in the Queue Managers folder context menu contains all possible actions for set management, as shown in Figure 8-13.

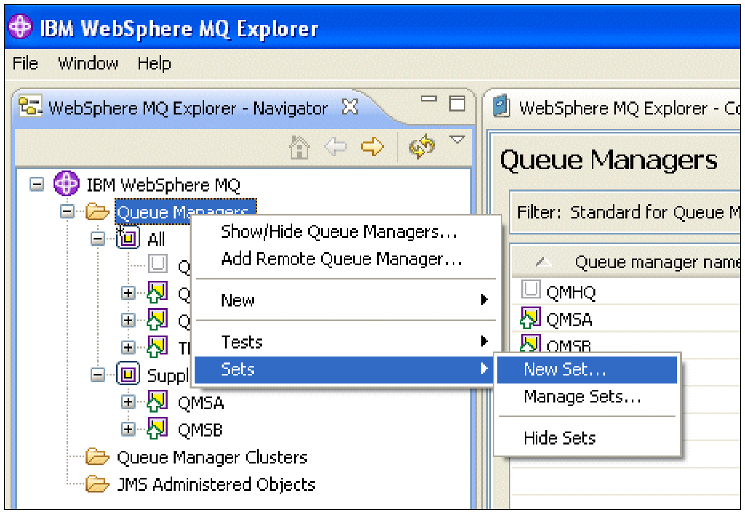


Figure 8-13 Context menu sets

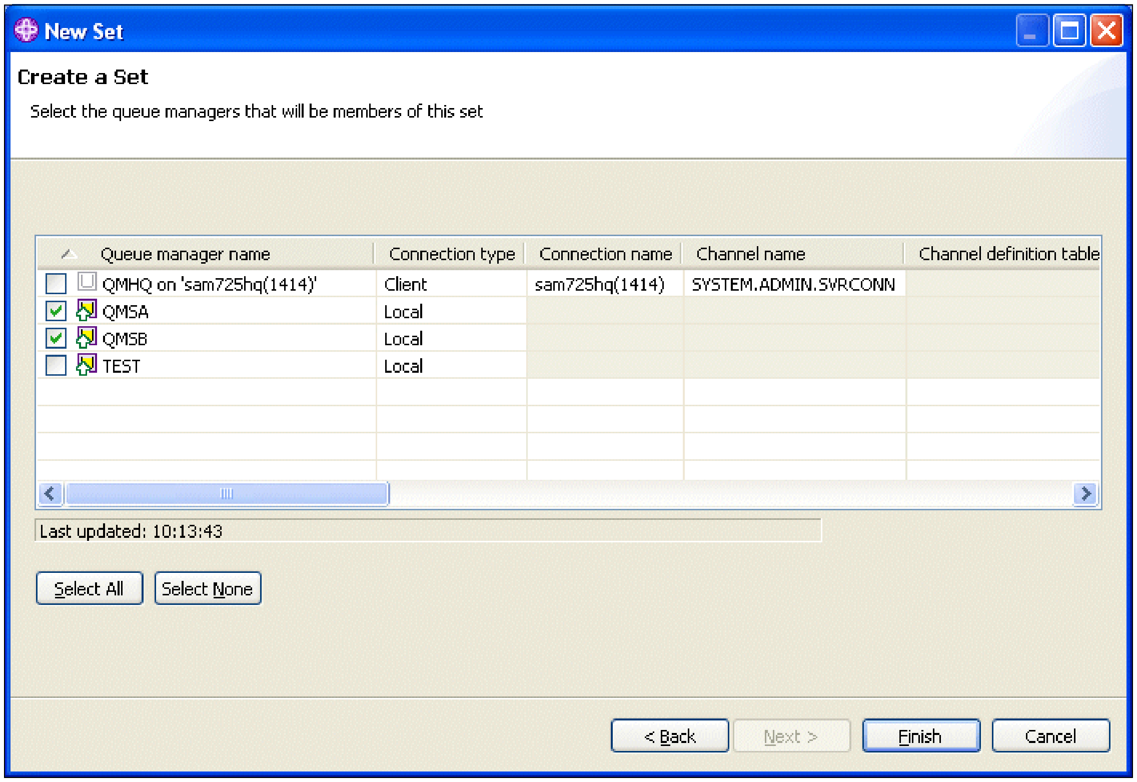
From this menu, it is possible to create a new set, manage existing sets, or show/hide all sets.

Creating a new set

To add a new set:

1. Select the **New Set** option from the Sets context menu.
2. In the first window enter a name for the new set. Click the appropriate radio button to select whether queue managers are to be added or removed from the set by manual actions or whether queue managers are to be automatically added or removed based on filter conditions. Once manual or automatic has been selected and the set has been created, this option cannot be changed.

3. In the next window, select queue managers (for a manual set) or filters (for an automatic set), as shown in Figure 8-14 and Figure 8-15 on page 176.



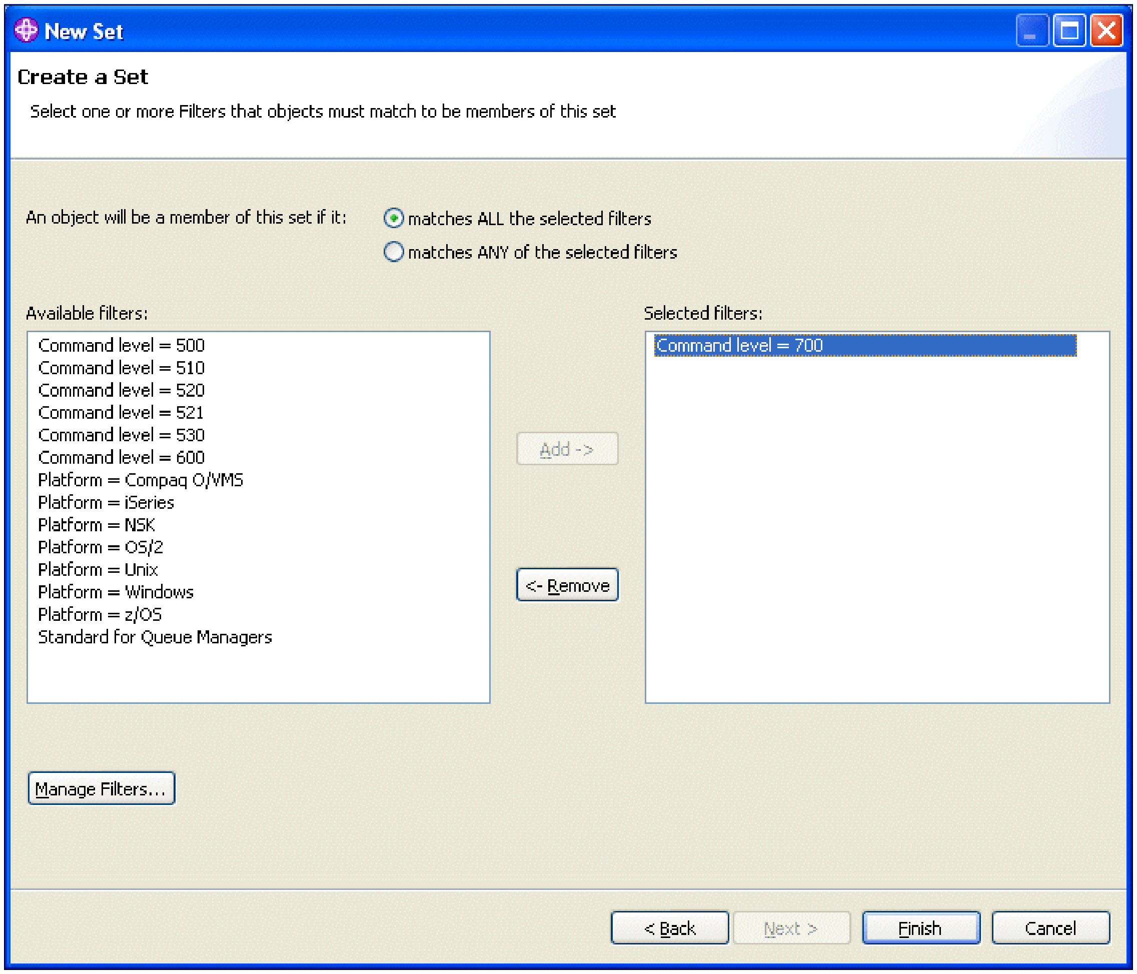
Create a Set

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


New 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

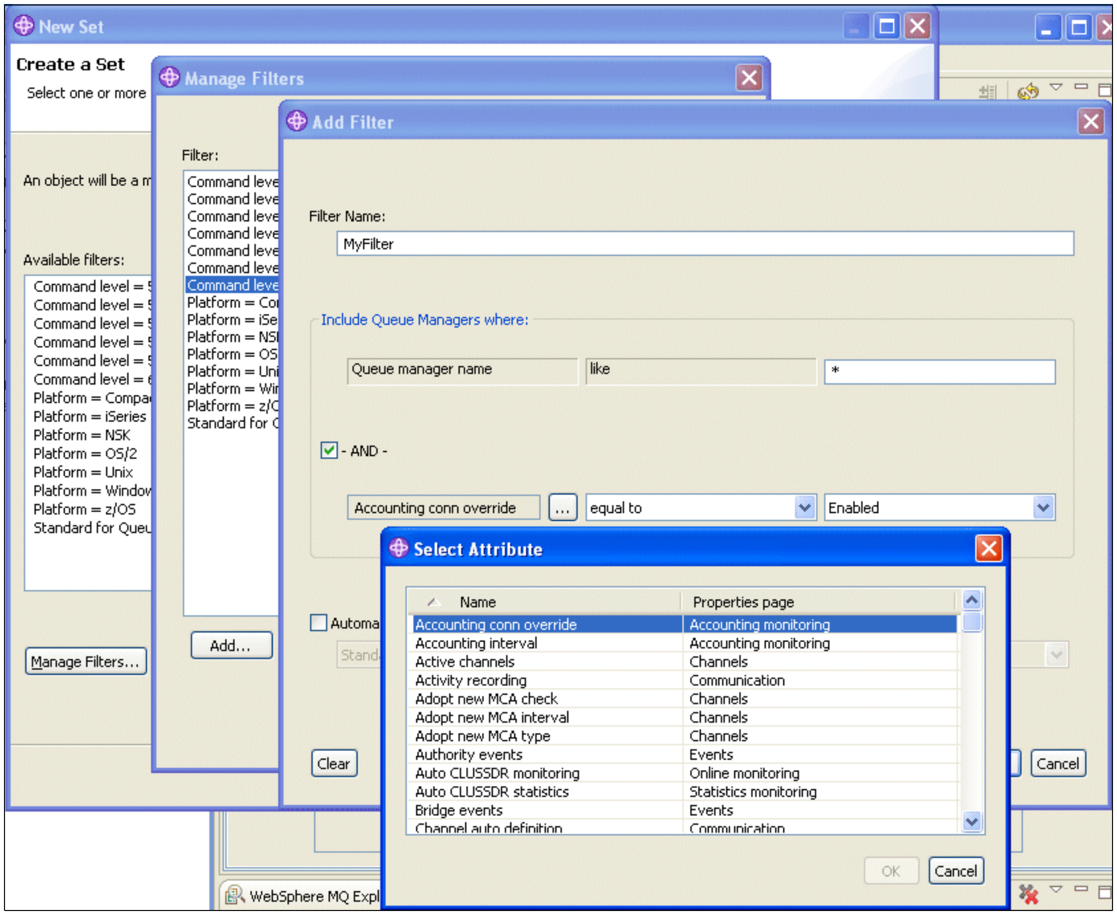
Selected filters:

Command level = 700

Figure 8-15 Filters selection for automatic set

4. In the case of a manual set, click **Finish**.

5. In case of automatic set, it is possible to define a custom filter. Click **Manage Filters** → **Add** to define the rules. To display a window that allows a custom filter to be based on queue manager attributes click the **...** (three dots) button, as shown in Figure 8-16.


New Set

Create a Set

Select one or more

An object will be a m

Available filters:

Command level = 5
Command level = 5
Command level = 5
Command level = 5
Command level = 5
Command level = 6
Platform = Compa
Platform = iSeries
Platform = NSK
Platform = OS/2
Platform = Unix
Platform = Window
Platform = z/OS
Standard for Queu

Manage Filters...

Manage Filters

Filter:

Command leve
Command leve
Command leve
Command leve
Command leve
Command leve
Command leve
Platform = Co
Platform = iSe
Platform = NS
Platform = OS
Platform = Uni
Platform = Wi
Platform = z/C
Standard for C

Add...

Add Filter

Filter Name: MyFilter

Include Queue Managers where:

Queue manager name	like	*

☑ - AND -

Accounting conn override	...	equal to	Enabled

☐ Automa

Stand

Clear

Cancel

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

WebSphere MQ Expl

Figure 8-16 Custom filter definition

6. Make a filter, then go back to the New Set wizard window and click **Finish**.

Using sets

The defined sets can be used for common administration tasks. Right-click the queue manager set name in the Navigator view and then select the desired action, as shown in Figure 8-17.

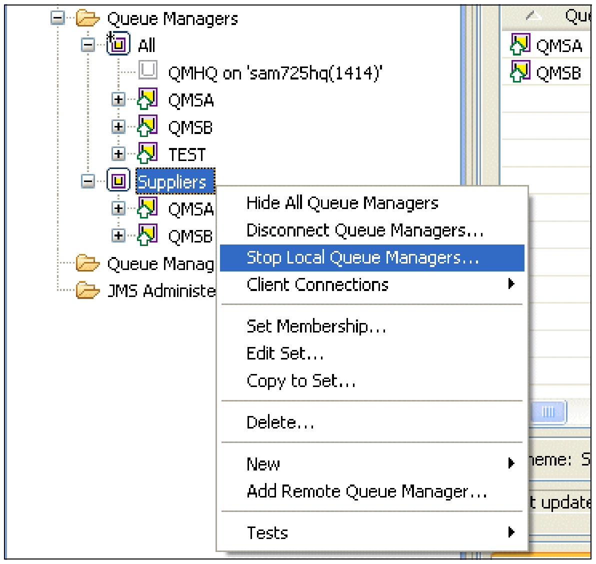


Figure 8-17 Queue manager set actions

Note: For further information about queue manager sets, refer to the topic Configuring WebSphere MQ Explorer → Creating and configuring a queue manager set in the MQ Explorer Help menu.

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?uid=swg24013508

8.2 Object properties and parameters

This section describes how to work with the new MQ object properties and parameters in WebSphere MQ V7.0. It discusses the WebSphere MQ Explorer and MQSC commands.

Note: In the context of this section, properties refer to fields that can be set on the MQ Explorer windows, not to properties of WebSphere MQ messages.

Note: On platforms other than z/OS, a parameter string that is specified as 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 (' ').

There are four parameters that are exceptions to this:

- TOPICSTR
- SUB
- USERDATA
- SELECTOR

For these parameters, two single quotation marks with no space are 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 queue manager parameters discussed in this section have been added.

Time interval for browsing message

The queue manager can automatically unmark messages that have been marked as browsed by a cooperating set of handles. If a marked message is not processed within the defined time interval, it is marked as not browsed.

To set this parameter:

- In MQ Explorer: Use the property called Message mark browse interval when changing or displaying queue manager properties.
- With MQSC: Issue commands ALTER QMGR or DISPLAY QMGR with parameter MARKINT.

The parameter value is in milliseconds, as an integer in the range 0 through 999 999 999. The default is 5000. There is a special value NOLIMIT that means that messages are not automatically unmarked.

Maximum size for message properties

This parameter specifies the maximum total length of properties data that can be associated with one WebSphere MQ message. To set this parameter:

- In MQ Explorer: Use the property called Max properties length in the Extended page when changing or displaying queue manager properties.
- With MQSC: Issue commands ALTER QMGR or DISPLAY QMGR with parameter MAXPROPL.

The parameter value is in bytes, as an integer in the range 0 through 104 857 600 bytes (100 MB). There is a special value NOLIMIT that means that there is no restriction on the size of message properties. The default is NOLIMIT.

Log defaults changes

Two changes have been made to logging parameters that can be set up for queue managers by control command **`crtmqm`** or in the qm.ini and mqs.ini files:

- The LogFilesPages default is now 4096. It was 1024 in the previous version of WebSphere MQ.
- The LogBufferPages default is now 512. It was 128 in the previous version of WebSphere MQ.

8.2.2 Queue object parameters

The parameters discussed in this section have been added to queue objects.

Queue alias

WebSphere MQ V7.0 introduces an extension to the queue alias object that allows an alias to be mapped to a topic object, as well as its previous usage of mapping to a Queue object. There are two new parameters on the queue alias object to support alias of a topic object or a Queue object.

A detailed description of using aliases to topics is provided in 4.2.3, “Topic alias” on page 47.

To set the new parameters:

- In MQ Explorer: The new property called base type can be assigned the value *queue* or *topic* to specify the type of the destination object. The new property called base object provides the name of the destination object (queue or topic). Base object replaces the base queue property that was used in MQ Explorer for WebSphere MQ V6.0.

  Both the base type and base object properties are in the General window when defining, changing, or displaying queue alias properties.
- With MQSC: Issue commands DEFINE QALIAS, ALTER QALIAS, and DISPLAY QUEUE with the new parameters TARGTYPE and TARGET.

  The destination (or target) object type is specified by TARGTYPE, and the value can be QUEUE or TOPIC. The destination (or target) object name is specified by TARGET. This replaces the TARGQ parameter used in previous versions of WebSphere MQ, although TARGQ has been retained in MQSC for WebSphere MQ V7.0 to provide backward compatibility for an alias to a Queue object.

Message properties

A message properties control parameter has been added in WebSphere MQ V7.0. This parameter determines how message properties are delivered to getting applications. It is applicable to local, alias, and model queue object types.

For details about message properties, refer to 6.2, “Message properties” on page 89.

To set this parameter:

- In MQ Explorer: Use the property called property control in the Extended window when defining, changing, or displaying queue properties.
- With MQSC: Issue commands **`DEFINE`** ***`queues`***, **`ALTER`** ***`queues`***, or **`DISPLAY`** ***`queues`*** with the parameter PROPCTL.

The parameter value can be:

- COMPAT (*Compatibility* in MQ Explorer)

  All message properties prefixed by mcd., jms., usr., or mqext. are delivered in MQRFH2. All other message properties are discarded. This is the default.
- ALL (*All* in MQ Explorer)

  All message properties are included in one or more MQRFH2 headers.

- FORCE (*Force MQRFH2* in MQ Explorer)

  Message properties are always returned in a MQRFH2 header, regardless of whether the application specifies a message handle. Properties are not accessible by message handle.

- NONE (*None* in MQ Explorer)

  All message properties are removed from the message.

Read ahead

A new parameter in WebSphere MQ V7.0 specifies the default read ahead behavior for non-persistent messages delivered to a client. Enabling read ahead can improve the performance of some client applications that consume 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, as read ahead is set for the SYSTEM.MQEXPLORER.REPLY.MODEL queue.

The read ahead feature is covered in detail in 5.4, "Read ahead" on page 68.

To set this parameter:

- In MQ Explorer: Use the property called Default read ahead in the Extended window when defining, changing, or displaying queue object properties.
- Using MQSC: 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 requests 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 any request by the client application.

Asynchronous put

A new parameter in WebSphere MQ V7.0 specifies the behavior to be used when an application specifies the put message options (MQPMO options) to include

MQPMO_RESPONSE_AS_Q_DEF. This parameter is applicable to local, alias, model, and remote queues.

The asynchronous put feature is described in 5.5, “Asynchronous put” on page 71.

To set this parameter:

- In MQ Explorer: Use the property called Default put response type in the Extended window when defining, changing, or displaying queue properties.
- Using MQSC: Issue commands **DEFINE *queues***, **ALTER *queues***, or **DISPLAY *queues*** with parameter DEFPRESP.

The parameter value can be:

- SYNC (*Sync* in MQ Explorer)

  This ensures that the put operations to the queue are issued in synchronous mode, as though the MQPMO_SYNC_RESPONSE option had been specified by the application. This is the default.
- ASYNC (*Async* in MQ Explorer)

  This ensures that the put operations to the queue are issued in synchronous mode, as though the MQPMO_ASYNC_RESPONSE option had been specified by the application.

8.2.3 New channel and client connection properties

The channel and client connection parameters discussed in this section have been added in WebSphere MQ V7.0.

Message properties

The message properties control attribute has been added. 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.

Message properties are described in 6.2, “Message properties” on page 89.

To set this parameter:

- In MQ Explorer: Use the property called Property control in the Extended window when defining, changing, or displaying channel properties.
- Using MQSC: 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 channel resources to limit the number of simultaneous running channels. This can prevent a single client application from monopolizing all channel resources on a queue manager. The situation could arise because:

- A program bug results in excessive connections to the queue manager
- A denial of service (DoS) program
- An unplanned roll-out of a client application to a large number of users

Details about server-connection resource control are discussed in 5.6, “Instance limits on SVRCONN channels” on page 76.

Maximum number of instances of server-connection channel

A new parameter sets the maximum number of simultaneous instances of an individual server-connection channel that can be started. To set this parameter:

- In MQ Explorer: Use the property called Max instances in the Extended windows when defining, changing, or displaying server-connection channel properties.
- Using MQSC: Issue commands DEFINE CHANNEL, ALTER CHANNEL, or DISPLAY CHANNEL with parameters CHLTYPE(SVRCONN) and MAXINST.

The parameter value is an integer in the range 0 to 999 999 999. The default is 999 999 999. The value 0 prevents all client access on the server-connections channel.

Maximum number of instances from a single client

A new parameter sets the maximum number of simultaneous server-connection channels that can be started from a single client (detected by IP address). To set this parameter:

- In MQ Explorer: Use the property called Max instance per client in the Extended window when defining, changing, or displaying server-connection channel properties.
- Using MQSC command: Issue commands DEFINE CHANNEL, ALTER CHANNEL, or DISPLAY CHANNEL with parameters CHLTYPE(SVRCONN) and MAXINSTC.

The parameter value is an integer in the range 0 to 999 999 999. The default is 999 999 999. The value 0 prevents all client access on the server-connection channel.

New error messages

When client connections reach the limits provided by the server-connection channel resource parameters the following error messages are written to the MQ error log:

- AMQ9489: The maximum number of instances was reached, as shown in Example 8-1.

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, as shown in Example 8-2.

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 indicate that there is a TCP/IP responder program error. Associated messages AMQ9489 and AMQ9489 describe the real reason for the error.

Note: On the Windows platform these messages are also written to the Windows Event log.

Conversation sharing

WebSphere MQ V7.0 now supports conversation sharing for client connections. A separate TCP/IP socket had to be used for each connection in the previous version of WebSphere MQ. Now it is possible to allow a number of connections to be shared on each TCP/IP channel instance, via one TCP/IP socket. This parameter is applicable to server-connection channel and client-connection channel definitions.

Details about conversations sharing are discussed in 5.3, “Conversation sharing” on page 64.

Setting the conversation sharing limit

To set this parameter:

- In MQ Explorer: Use the property called Sharing Conversations in the Extended windows when defining, changing, or displaying server-connection channels or client connection channels.
- Using MQSC: Issue commands DEFINE CHANNEL, ALTER CHANNEL, or DISPLAY CHANNEL with parameters CHLTYPE(SVRCONN) or (CLNTCONN) and SHARECNV.

The parameter value is an integer in the range 0 to 999 999 999. The default is 10. There are two special values:

- 1

  Sharing of conversations is disabled. The channel runs in V7 mode and the read ahead and asynchronous put features are still available.

- 0

  Sharing of conversations is disabled. The channel runs in pre-V7 mode and the read ahead and asynchronous put features are not available.

Displaying the conversation sharing parameters

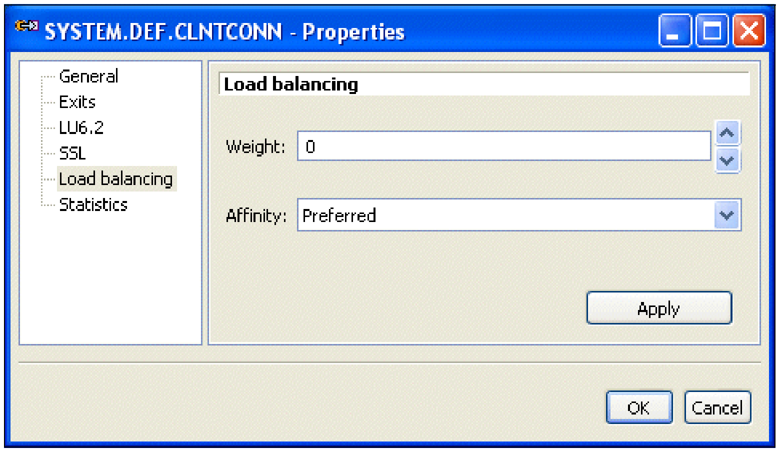
To display these parameters:

- In MQ Explorer: Right-click a server-connection channel, select **Status** → **Channel Status**, and refer to the properties max conversations and current conversations. These are the last two columns in the list.
- Using MQSC: Issue the command DISPLAY CHSTATUS with parameters MAXSHCNV and CURSHCNV.

A zero (0) displayed value for these parameters indicates that the 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 related to this feature. In MQ Explorer, both parameters can be set from the new Load balancing window when working with the client connections wizard, as shown in Figure 8-18.


SYSTEM.DEF.CLNTCONN - Properties

Load balancing

Weight: 0

Affinity: Preferred

Figure 8-18 Client connection load balancing

See 5.7, “Weighted selection on CLNTCONN channels” on page 78, for a description of client-connection channel load balancing using the weight and affinity parameters.

Client connection affinity

The affinity attribute is used to control how client applications select a channel to connect to a queue manager. This parameter is only applicable when multiple channel definitions are available. To set this parameter:

- In MQ Explorer: Use the property called affinity in the Load balancing window when defining, changing, or displaying client-connection channel properties.
- Using MQSC: Issue command 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 host name 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 weighting attribute is used so that client channel definitions can be selected at random based on their weighting when more than one suitable definition is available. To set this parameter:

- In MQ Explorer: Use the property called weight in the Load balancing window when defining, changing, or displaying client-connection channel properties.
- Using MQSC: Issue command DEFINE CHANNEL, ALTER CHANNEL, or DISPLAY CHANNEL with parameters CHLTYPE(CLNTCONN) and CLNTWGHT.

The parameter value is an integer in the range of 0 to 99. The default is 0. The value 0 indicates that no random load balancing is performed and applicable definitions are selected in alphabetical order.

8.2.4 Connection status and queue status enhancements

It is now possible to display additional information about applications connected to the queue manager and the queues that they are using.

The connection status parameters discussed in this section have been added.

Display connection status for read ahead

This shows the state of read ahead on a connection handle. Details about read ahead are discussed in 5.4, “Read ahead” on page 68.

To display connection status:

- In MQ Explorer: Right-click a queue manager and select **Application Connections**. Then in the status window refer to the value read ahead in the lower section of this window.
- Using MQSC: Issue the command DISPLAY CONN with parameter READA.

The displayed value can be:

- NO (*No* in MQ Explorer)

  Read ahead of non-persistent messages is not enabled for this connection.
- YES (*Yes* in MQ Explorer)

  Read ahead of non-persistent messages is enabled for this connection 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 connection. Read ahead is not being used efficiently because the client has been sent a large number of messages that have not been consumed.

Display connection status for asynchronous consume

This shows the state of asynchronous consumption on connection handles. Asynchronous consumption (callback) is described in 6.4, “Callback for asynchronous consumers” on page 108.

To display connection status:

- In MQ Explorer: Right-click a queue manager and select **Application Connections**. Then in the status window refer to the value asynchronous state in the lower section of this window.
- Using MQSC: Issue the command DISPLAY CONN with parameter ASTATE.

The displayed value can be:

- SUSPENDED (*Suspended* in MQ Explorer)

  The asynchronous message consumption is temporarily suspended on this connection. A call to MQCTL with MQOP_SUSPEND has been issued against the connection handle.
- STARTED (*Started* in MQ Explorer)

  The asynchronous message consumption can proceed on this connection. A call to MQCTL with MQOP_START has been issued against the connection handle.
- STARTWAIT (*Startwait* in MQ Explorer)

  The asynchronous message consumption can proceed on this connection. A call to 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. A call to MQCTL 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 call to MQCTL has been issued against the connection handle.

Display queue status for asynchronous consume

This shows the state of asynchronous consumption on queue objects.

To display queue status:

- In MQ Explorer: Right-click a queue and select **Status**. Then in the status window refer to the value asynchronous state in the lower section of this window.
- Using MQSC: Issue the command DISPLAY QSTATUS with parameter ASTATE.

The displayed value can be:

- ACTIVE (*Active* in MQ Explorer)

  Asynchronous message consumption is running on this object handle. A call to MQCB has set up a callback function to process messages asynchronously.

- INACTIVE (*Inactive* in MQ Explorer)

  The connection handle has not yet been started, or has been stopped or suspended. A call to MQCB has set up a callback function to process messages asynchronously.

- SUSPENDED (*Suspended* in MQ Explorer)

  The asynchronous message consumption is suspended on this object handle. This can be either because a MQCTL call with MQOP_SUSPEND has been issued by the application, or because it has been suspended by WebSphere MQ.

- SUSPTEMP (*Susptemp* in MQ Explorer)

  Asynchronous message consumption is temporarily suspended by WebSphere MQ.

- NONE (*None* in MQ Explorer)

  Asynchronous message consumption cannot currently proceed on this connection. No call to MQCTL has been issued against the connection handle.

Important: Only a short explanation of properties and parameters for MQ Explorer and MQSC commands has been provided in this section. For complete details about properties and parameters, refer to the topic Reference → Properties in the MQ Explorer Help menu and the manual *Script (MQSC) Command Reference* available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

To use PCF commands, refer to the manual *Programmable Command Formats and Administration Interface* in the WebSphere MQ V7.0 Information Center.

8.3 Java and JMS-related administration enhancements

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

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

8.3.1 Embedded PCF support for Java

Support for creating WebSphere MQ message headers and messages containing PCF commands is now contained in the file com.ibm.mq.headers.jar that is part of the 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. For details about this SupportPac refer to the IBM Web page at:

`http://www.ibm.com/support/docview.wss?uid=swg24000668`

Note: Programmable Command Format (PCF) allows administration tasks to be written into an application program by sending special messages to the WebSphere MQ Command Server. PCF commands cover the same range of functions provided by MQSC commands.

For detailed information refer to the manual *Programmable Command Formats and Administration Interface* in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

8.3.2 WebSphere MQ classes for JMS

WebSphere MQ classes for JMS has been enhanced to provide a 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 that contains plain text messages about errors that must be corrected. 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 assists IBM Service to diagnose problems.
- 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.

Refer to Chapter 7, "WebSphere MQ Java Message Service enhancements" on page 137, for further details.

8.4 Control commands

This section describes new control commands in WebSphere MQ V7.0 and existing control commands that have new parameters.

Important: Only a only short explanation of the parameters for WebSphere MQ control commands is provided here. For complete details refer to the *System Administration Guide* and *i5/OS System Administration Guide*, available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

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 it runs as a service. This is the default option when a queue manager is created from MQ Explorer.
- **`-si`**: Interactive (manual) queue manager startup

  The queue manager is configured to start only when the **`strmqm`** command is issued manual. It then runs under the logged on (interactive) user and ends when the user who started it logs off.
- **`-ss`**: Service (manual) queue manager startup

  The queue manager is configured to start only when the **`strmqm`** command is issued manually. It then runs as a service and continues to run even after the interactive user has logged off. This is the default option when a queue manager is created by the **`crtmqm`** command.

The command syntax for the new parameters can be:

- **`crtmqm -sa`** ***`queue_manager_name`***
- **`crtmqm -si`** ***`queue_manager_name`***
- **`crtmqm -ss`** ***`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 it logs off.

- **`-ss`**: Service (manual) queue manager startup

  The queue manager runs as a service and continues 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 the new options are not specified the startup type is as set previously by one of these methods.

The command syntax for the new parameters can be:

- **`strmqm -si`** ***`queue_manager_name`***
- **`strmqm -ss`** ***`queue_manager_name`***

8.4.3 WebSphere MQ CL commands on i5/OS

The following CL commands are new for WebSphere MQ V7.0 on the i5/OS (iSeries®) platform to administer topics and subscriptions:

- **`CHGMQMTOP`**
- **`CLRMQMTOP`**
- **`CPYMQMTOP`**
- **`CRTMQMTOP`**
- **`DLTMQMSUB`**
- **`DSPMQMSUB`**
- **`WRKMQMSUB`**

The following CL commands have been enhanced for new parameters:

- CRTMQM to accommodate JRNBUFSIZ
- CRTMQMQ, CHGMQMQ, and CPYMQMQ to accommodate MSGREADAHD, DFTPUTRESP, PROPCTL, and TARGTYPE

- GRTMQMAUT and RVKMQMAUT to accommodate OBJTYPE(*TOPIC)
- RCDMQMIMG and RCRMQMOBJ to accommodate OBJTYPE(*TOPIC)

8.5 Journals on i5/OS

WebSphere MQ V7.0 for the i5/OS (iSeries) platform uses a time-stamping technique that is different from earlier versions of WebSphere MQ for i5/OS. This solves the issue of duplicate time stamps in journals at the end of Daylight Saving Time (DST). The administrator need not intervene to change the clock during DST. The queue manager can run uninterrupted through a DST time change. The journal buffer size can also now be larger than the default 32kb on this platform.

9

Publish/Subscribe management

The WebSphere MQ v7.0 Publish/Subscribe environment can be managed using MQ Explorer or MQSC. This chapter shows how to use these administration tools to set up a Publish/Subscribe environment, manage it, and display its status. This chapter contains the following sections:

- 9.1.1, “Creating topics using MQ Explorer” on page 198, and 9.1.2, “Creating topics using MQSC” on page 202
- 9.1.3, “Altering topics using MQ Explorer” on page 204, and 9.1.4, “Altering topics using MQSC” on page 205
- 9.1.5, “Displaying topic status using MQ Explorer” on page 206, and 9.1.6, “Displaying topic status using MQSC” on page 207
- 9.1.7, “Creating JMS topics using MQ Explorer” on page 208
- 9.1.8, “Setting up topic security using MQ Explorer” on page 213, and 9.1.9, “Setting up topic security using setmqaut” on page 215
- 9.1.10, “Mapping queue aliases to a topic object” on page 215
- 9.2, “Managing subscriptions” on page 216, and 9.3, “Displaying Publish/Subscribe status” on page 221

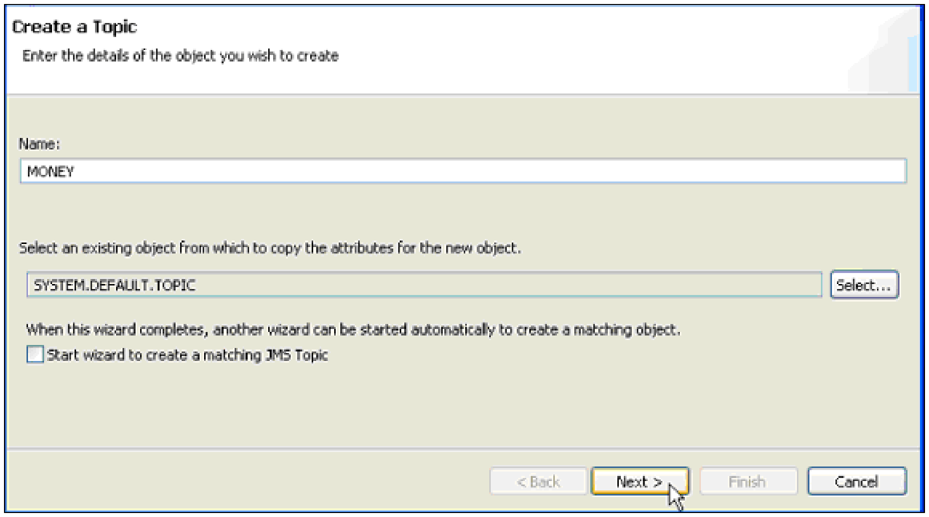
9.1 Managing topics

This section illustrates the usage of MQ Explorer and MQSC commands for managing topics in a Publish/Subscribe environment.

9.1.1 Creating topics using MQ Explorer

The following procedure uses MQ Explorer on the Windows platform to create a topic object on a local or remote queue manager:

1. Navigate to the Topics folder in a queue manager.
2. Right-click the **Topics** folder and select **New - Topic** from the context menu. The Create a Topic window appears (Figure 9-1).



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

3. Enter the name for the topic object that must be created and click **Next**.

4. The Change Properties window appears (Figure 9-2). A topic string must be entered in the Topic String text field. This topic string signifies the topic hierarchy. The Description text field is optional.

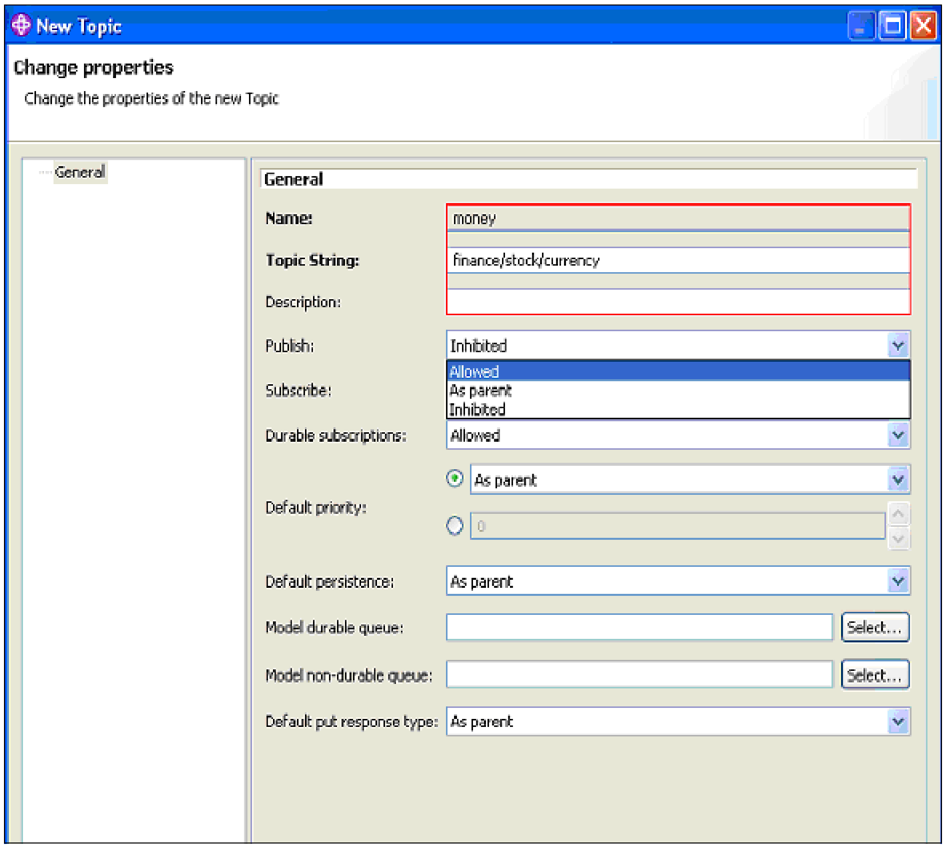
   The publish property can take on three values:

   – 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, the default may have been changed since installation.

   – Inhibited: Messages cannot be published to this topic.

   The Subscribe property can also take on three values: Allowed, As Parent, and Inhibited. These values have the same meaning as in the publish property, except they are for the context of subscription.


New Topic

Change properties

Change the properties of the new Topic

General

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

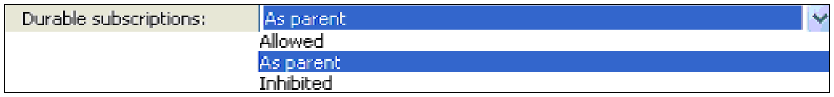
The Durable Subscriptions property can take on three values:

– 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 may have been changed since installation.

– Allowed: Durable subscriptions can be made on this topic.

– Inhibited: Durable subscriptions cannot be made on this topic.

Refer to Figure 9-3.



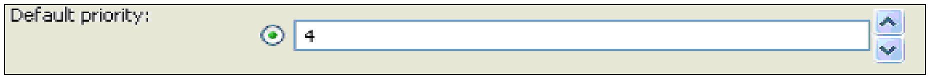
Durable subscriptions: As parent

Allowed

As parent

Inhibited

Figure 9-3 Configuring durable subscriptions



Default priority: 4

Figure 9-4 Configuring default priority

The default priority property can be set to assign a default priority on messages published to that topic. Refer to Figure 9-4. The value must be in the range zero (the lowest priority) through to the MAXPRTY queue manager parameter (MAXPRTY is 9). If the default priority is set to the special value *as parent,* 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. However, this may have been changed since installation.

The default persistence property specifies the persistence of published messages if the publisher application does not specify it as persistent or non-persistent:

– As Parent: 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 may have changed since installation.

– Not Persistent: Messages are lost during a restart of the queue manager.

– Persistent: Messages survive a queue manager restart.

Refer to Figure 9-5.

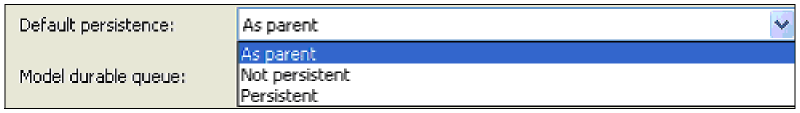

Default persistence: As parent

Figure 9-5 Configuring default persistence

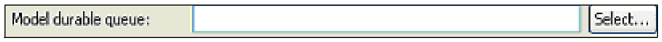


Figure 9-6 Configuring the model durable queue

The model durable queue property can be used to specify the name of a queue model object for durable subscriptions that request the queue manager to manage the destination of its publications. The maximum length of 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. Refer to Figure 9-6.

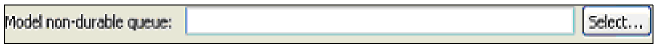


Figure 9-7 Configuring the model non-durable queue

The model non-durable queue property can be used to specify the name of a queue model object for non-durable subscriptions that request that the queue manager manage the destination of its publications. The maximum length of 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. Refer to Figure 9-7.

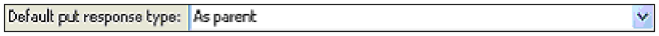

Default put response type: As parent

Figure 9-8 Configuring default-put response type

The default put response type property is a means of administratively configuring the topic object to support synchronous/asynchronous methods 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 to with a Completion Code. WebSphere MQ v7.0 introduces the asynchronous put response type by selecting Async. Further details on concepts involved and using this feature in WebSphere MQ v7.0 are discussed in Chapter 6, “Message Queue Interface extensions” on page 85. *As parent* is the default value for this property and has the same meaning that it has on all other properties. Refer to Figure 9-8 on page 201.

Having configured all these properties, out of which only the topic name and the topic string are required, the creation of the topic object can be completed by clicking **OK**.

9.1.2 Creating topics using MQSC

The following commands provide simple examples of using the MQSC administration interface to create topic objects.

1. Start the MQSC interface by entering the command:

   ```
   runmqsc queue_manager_name
   ```

2. Enter:

   ```
   DEFINE TOPIC(DELI) TOPICSTR(‘deli’)
   ```

   This creates a topic object with the name “DELI”, which is an administrative node in the topic hierarchy for the topic string “deli”.

3. Enter:

   ```
   DEFINE TOPIC(DELI.FRESH) TOPICSTR('deli/fresh')
   ```

   This creates a topic object “DELI.FRESH” that is a child administration node of the topic object “DELI” created previously, by virtue of the hierarchy specified in the topic string “deli/fresh”. By default it inherits all attributes from the immediate parent topic object.

4. Enter:

   ```
   DEFINE TOPIC(MATTS) TOPICSTR('deli')
   ```

   This results in the error:

   ```
   A WebSphere MQ Topic using the supplied topic string already exists.
   ```

   This is because this attempted topic definition conflicts with the topic object “DELI”. Two administrative topic objects cannot be created at the same node in the topic string hierarchy.

The new parameter options available for the DEFINE TOPIC command in MQSC are the same as those discussed on the Change Properties window of MQ Explorer described in the previous sub-section.

The MQSC parameters have different names for the properties on the MQ Explorer windows. A mapping between the MQSC attributes and the corresponding MQ Explorer Properties is listed in Table 9-1.

Table 9-1 MQ Explorer topic creation properties corresponding to MQSC DEFINE TOPIC parameters

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 in range 0 to 9
Default Persistence	DEFPSIST	ASPARENT YES NO
Model Durable Queue	MDURMDL	Model Object name
Model Non Durable Queue	MNDURMDL	Model Object name

There are other parameters that can be used with the DEFINE TOPIC command in MQSC. For more information about to this command refer to the manual *Script (MQSC) Command Reference* available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

9.1.3 Altering topics using MQ Explorer

The following procedure uses MQ Explorer on the Windows platform to alter a topic object on a local or remote queue manager:

1. Click the **Topics** folder of a queue manager to display the list of topic object names that exist on the queue manager.
2. Right-click a topic object. This displays the context menu shown in Figure 9-9. Select **Properties** from the context menu. This displays the **Topic Properties** window, as shown in Figure 9-10 on page 205.

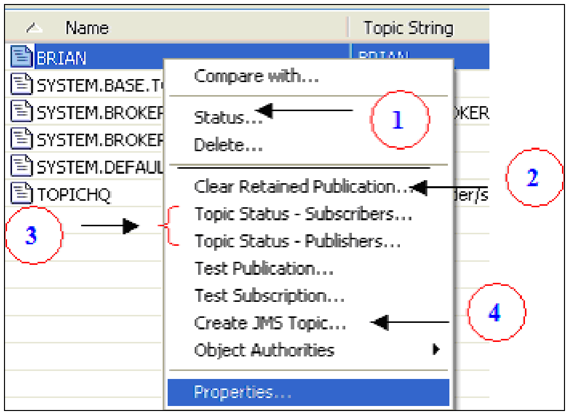
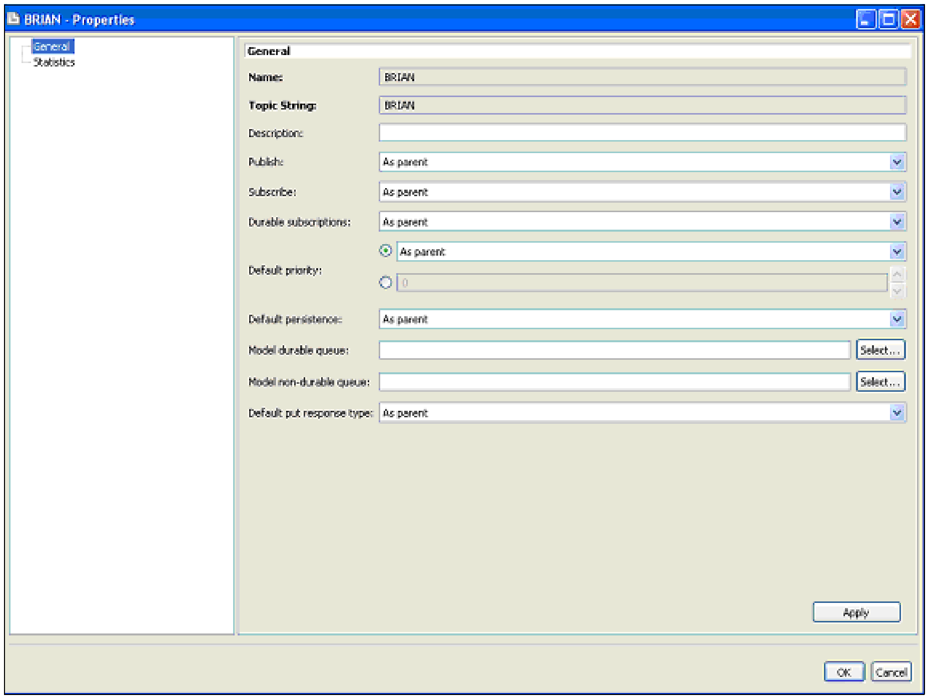


Figure 9-9 Topic menu

3. The properties listed here are the same as those listed on the Change Properties window, which is discussed in 9.1.1, “Creating topics using MQ Explorer” on page 198. Alter the properties as desired and then click **Apply** for the changes to take effect immediately and leave the window open, or click **OK** for the changes to take effect and close the window.



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

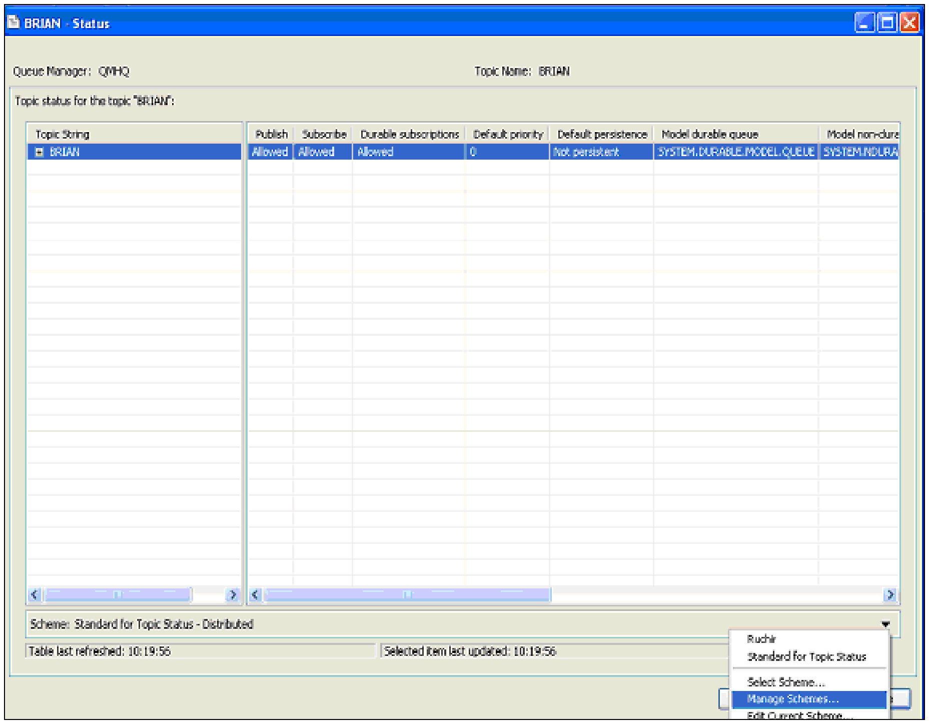
9.1.4 Altering topics using MQSC

To alter topic objects, WebSphere MQ v7.0 provides the MQSC command ALTER TOPIC. The parameters are the same as used with the DEFINE TOPIC command. Refer to 9.1.2, “Creating topics using MQSC” on page 202, for details.

9.1.5 Displaying topic status using MQ Explorer

The following procedure uses MQ Explorer on the Windows platform to display the status of a topic on a local or remote queue manager:

1. Click the **Topics** folder of a queue manager to display the list of topic object names that exist on the queue manager.
2. Right-click a topic object. This displays the context menu shown in Figure 9-9 on page 204. Select **Status** from the context menu, as highlighted by a 1 in Figure 9-9 on page 204. This displays the Topic Status window, as shown in Figure 9-11. This reports the current values for all properties of the topic.
3. If only a given subset of the properties must be displayed, MQ Explorer provides an option to create a scheme. Right-click **Scheme Panel** and then click **Manage Schemes**. It is then possible to define a named scheme that only consists of the properties that must be viewed.



Queue Manager: QMHQ　　Topic Name: BRIAN

Topic status for the topic "BRIAN":

Topic String
BRIAN

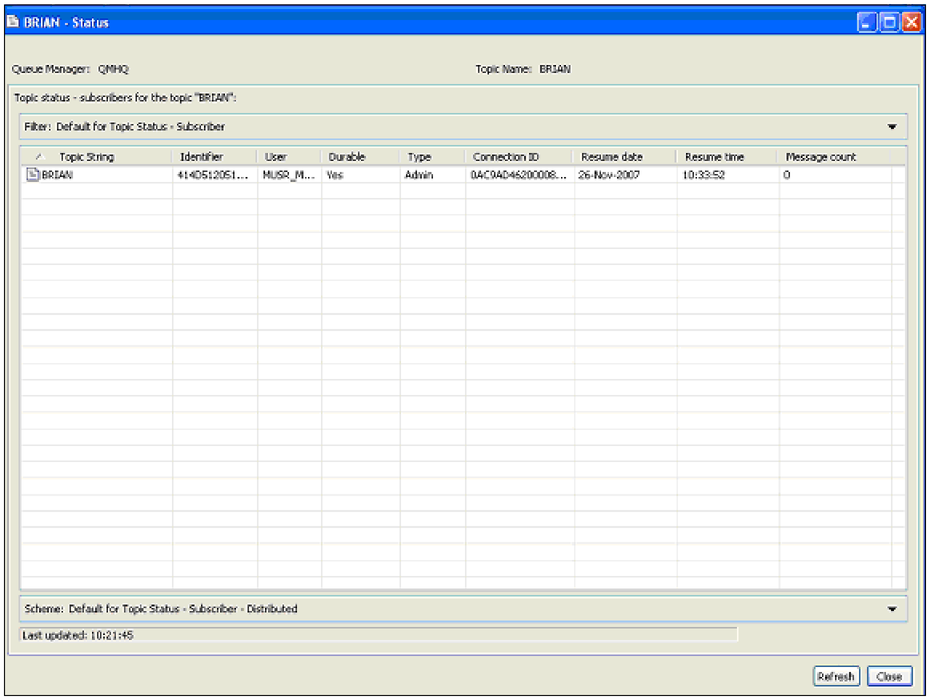
Publish	Subscribe	Durable subscriptions	Default priority	Default persistence	Model durable queue	Model non-dura
Allowed	Allowed	Allowed	0	Not persistent	SYSTEM.DURABLE.MODEL.QUEUE	SYSTEM.NDURA

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

4. Right-click a topic name in MQ Explorer and then click **Topic Status - Publishers and Topic Status - Subscribers**, marked as 3 in Figure 9-9 on page 204. This opens windows that displays the status of publications and subscriptions, respectively. Figure 9-12 shows an example of displaying the status of subscribers.



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 MQSC command DISPLAY TPSTATUS is used to display the status of one or more topic nodes in a topic tree as determined by the supplied parameters. The syntax and three examples are provided in Example 9-1.

Example 9-1 Displaying topic status using TPSTATUS command

```
DISPLAY TPSTATUS(topic_string) WHERE(filter_condition)
   TYPE(TOPIC|PUB|SUB) ALL

DISPLAY TPSTATUS(‘deli/fresh/fruit’) TYPE(TOPIC) ALL
DIS TPS(‘deli/fresh/#’) TYPE(PUB) ALL
DIS TPS(‘deli/+’) WHERE(PUBCNT GT 0)TYPE(TOPIC) ALL
```

The DISPLAY TPSTATUS command requires a topic string to determine which topic nodes are displayed. The WHERE keyword specifies a filter condition to display only those administrative topic definitions that satisfy the selection criterion of the filter condition. A list all other parameters that can be used and displayed is found in the manual *Script (MQSC) Command Reference*, available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

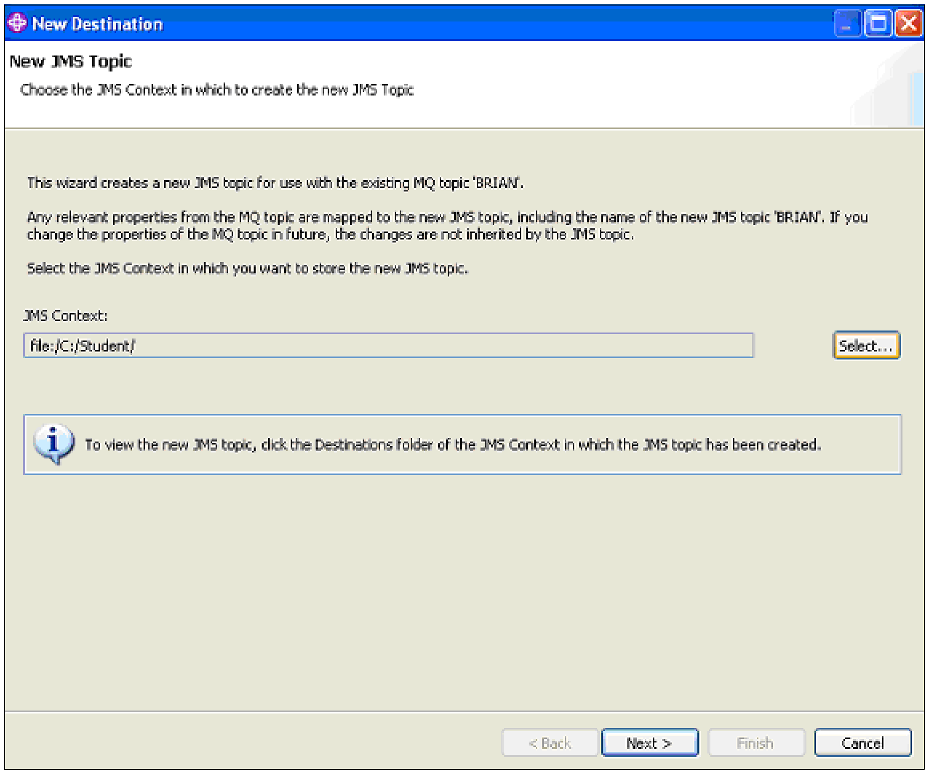
The parameter ALL can be used to display the values of all properties of a topic. The topic status display can also be wildcarded, for example, DISPLAY TPSTATUS ('#'). The hash (#) matches any topic string and therefore the command displays values of properties of all topic objects corresponding to each topic string currently defined in the system.

9.1.7 Creating JMS topics using MQ Explorer

The following procedure uses MQ Explorer on the Windows platform to display the status of a topic on a local or remote queue manager:

1. The JMS Context is a prerequisite for creating JMS Topics. The JMS Context can be created by right-clicking **JMS Administered Objects** in the Navigator view of MQ Explorer and selecting **Add Initial Context** from the context menu. Valid JNDI name space, security options, and connection details must be entered. Click **Finish** to complete the creation of the JMS Context.
2. Click the **Topics** folder of a queue manager to display the list of topic object names that exist on the queue manager.

3. Right-click a topic name and then select **Create a JMS Topic** from the context menu, marked as 4 in Figure 9-9 on page 204. This displays a window for creating a JMS Topic that is the JMS equivalent of the selected MQ Topic, as shown in Figure 9-13.



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.

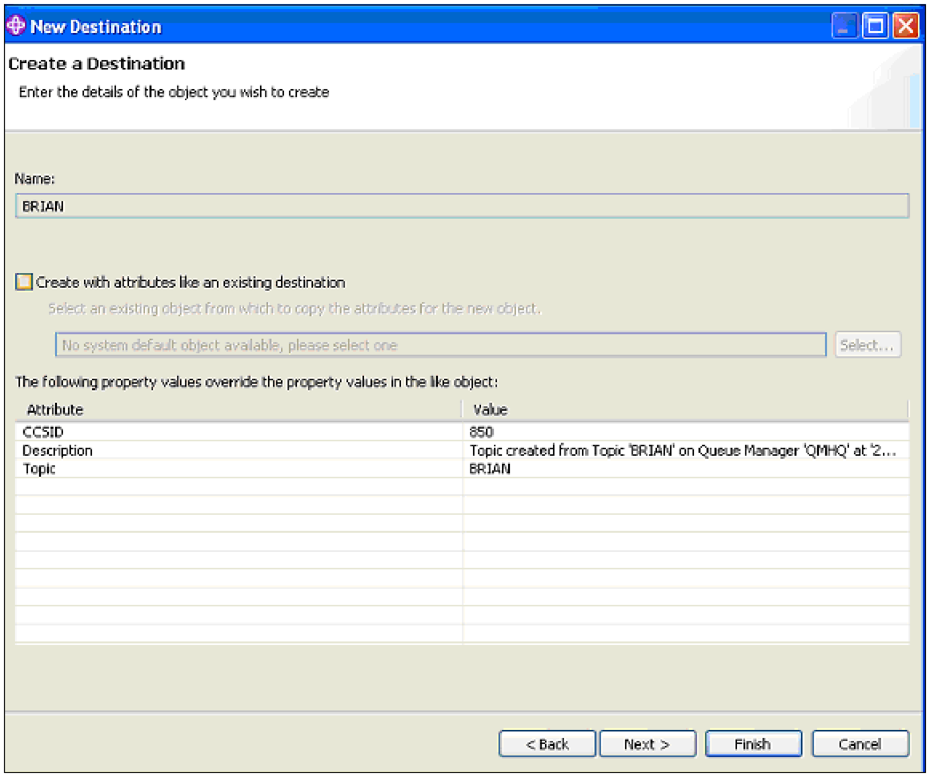
Select the JMS Context in which you want to store the new JMS topic.

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

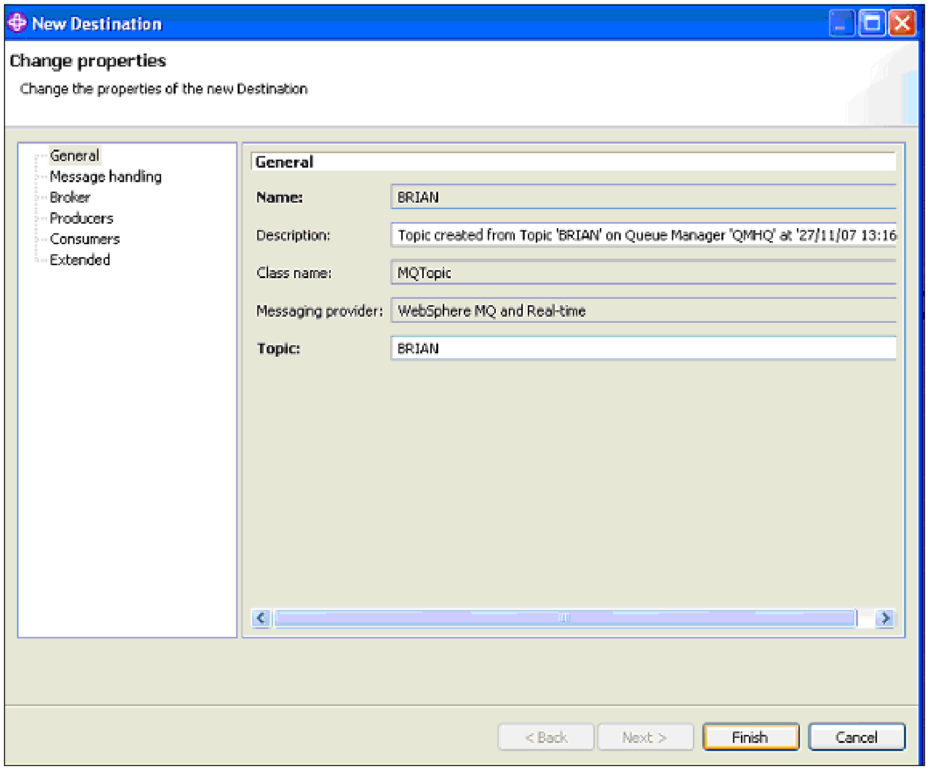
4. Figure 9-14 shows that it is possible to create a JMS destination with attributes like an existing destination. Click **Finish** to complete the creation of the JMS Topic or click **Next** to go to the Change Properties window, as shown in Figure 9-15 on page 211.


New Destination
Create a Destination
Enter the details of the object you wish to create
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
Select...
< Back
Next >
Finish
Cancel

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 preexisting JMS Topic



Change properties

Change the properties of the new Destination

General

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

5. On the left side of the windows there are categories of properties that can be changed on the JMS Topic:

   – The Message Handling category has properties such as Expiry, Persistence, and Priority that relate to how messages are handled by MQ. Expiry can have the value Application, which means that it can be set inside the JMS Application, or Unlimited, which means that messages have no expiry, or a user-specified value.

     Persistence can have value 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 set to High, both Persistent and Non Persistent would mean the same as they do in the WebSphere MQ V6.0 queue default, meaning that default persistence level is set for the queue that holds the messages. Priority means the same thing as does in WebSphere MQ V6.0.

   – The Broker category has properties such as Broker Durable Subscription Queue, which specifies the queue that holds messages for durable subscriptions for JMS-based clients. The Broker CC Durable Subscription Manager Queue serves the same function as the Broker Durable Subscription Queue except that it caters for JMS 1.1 based clients. The publication stream refers to the stream name. Streams provide a means 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, which is 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. 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 category has the asynchronous puts property 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.

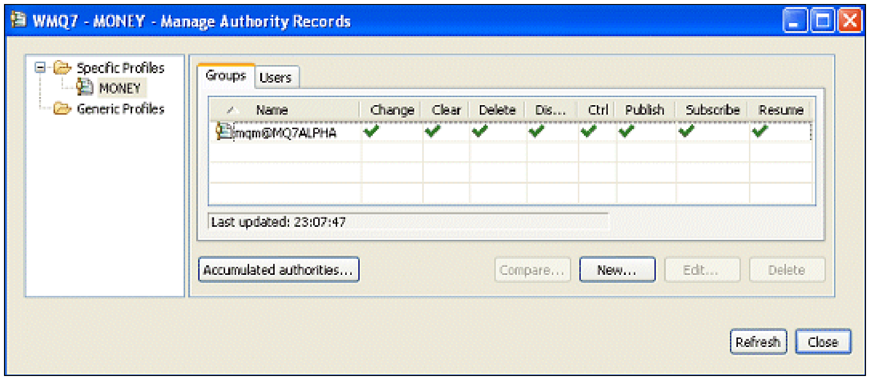
– The consumers category has the allow read ahead property that can have values As Destination, Enabled, or Disabled. These have the same meanings as for asynchronous puts in the producer category.

9.1.8 Setting up topic security using MQ Explorer

The following procedure uses MQ Explorer on the Windows platform to set up security for a topic object on a local or remote queue manager:

1. Click the **Topics** folder of a queue manager to display the list of topic object names that exist on the queue manager.
2. Right-click a topic name and then select **Object Authorities - Manage Authority Records** from the context menu, as shown in Figure 9-9 on page 204.

   This opens the Manage Authority Records window, as shown in Figure 9-16. 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	✓	✓	✓	✓	✓	✓	✓	✓

Last updated: 23:07:47

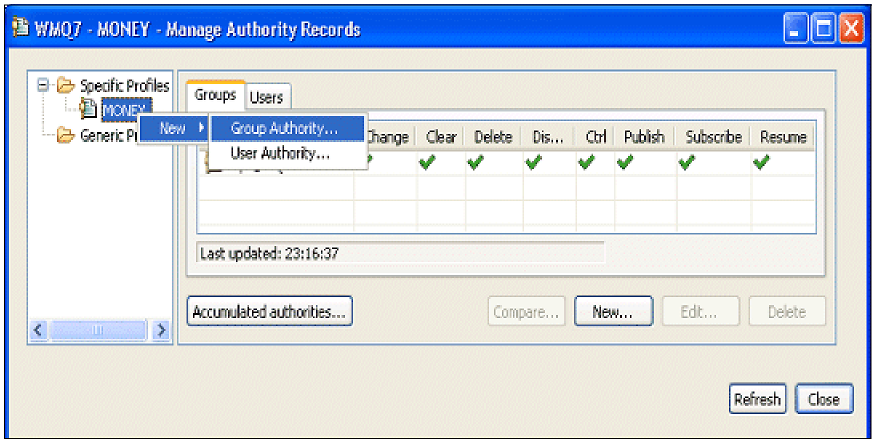
Figure 9-16 Managing specific profiles

This window 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, 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 have been added to support security for Publish/Subscribe in WebSphere MQ v7.0. These three new authorities are:

- Publish: Enabling this authority means that users in the group can publish to the topic using the MQPUT call.
- Subscribe: Enabling this authority means that users in the group can create, alter, or resume a subscription to a topic using the MQSUB call.
- Resume: Enabling this authority means that users in this group can resume a subscription using the MQSUB call.

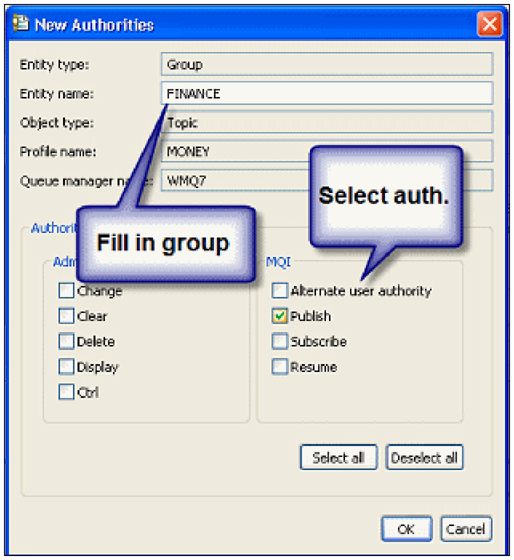
It is possible to create a group authority or user authority, as illustrated in Figure 9-17.



Last updated: 23:16:37

Figure 9-17 Creating GROUP Authority FINANCE

3. Right-click the specific profile **MONEY** and then click **New - Group Authority**. This opens the New Authorities window, as shown in Figure 9-18.



Entity type: Group
Entity name: FINANCE
Object type: Topic
Profile name: MONEY
Queue manager name: WMQ7

Admin:
- ☐ Change
- ☐ Clear
- ☐ Delete
- ☐ Display
- ☐ Ctrl

MQI:
- ☐ Alternate user authority
- ☑ Publish
- ☐ Subscribe
- ☐ Resume

Figure 9-18 Setting new authorities for group FINANCE

A range of authorities can be enabled for the new group. Clicking **OK** creates a new group authority with the selected authorities.

9.1.9 Setting up topic security using setmqaut

The **setmqaut** control 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 principals or groups to which the authorizations apply must be specified, along with the queue manager, object type, and the profile name that identifies the objects. Example 9-2 demonstrates the use of **setmqaut** with the topic type. It also shows usage of the three new kinds of authorities:

- Pub
- Sub
- Resume

Example 9-2 Using setmqaut

```
To let the users in the group “editors” subscribe to topic “DELI” on
Queue Manager WMQ7:
setmqaut -m WMQ7 -n DELI -t topic -g editors +sub +resume

And to allow users in the group “journalists” to publish to the topic:
setmqaut -m WMQ7 -n DELI -t topic -g journalists +pub
```

Note: -pub and -sub clear the authority to publish and subscribe on the topic. If a group has a +sub for a parent administration topic node and -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 administration nodes.

For a complete description of the options that can be used with the **setmqaut** command refer to the manual *System Administration Guide*, available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

9.1.10 Mapping queue aliases to a topic object

The queue alias object has been extended in WebSphere MQ V7.0 to allow aliases to be created for the new topic object. Refer to 4.2.3, “Topic alias” on page 47, for a complete description of this enhancement and its usage.

The administration procedure for MQ Explorer and MQSC is given in 8.2.2, “Queue object parameters” on page 180.

9.2 Managing subscriptions

WebSphere MQ v7.0 provides for management of administrative subscriptions through MQ Explorer and MQSC. This allows point-to-point applications to act as subscribers without any code changes to the applications, as described in 4.2.3, “Topic alias” on page 47.

9.2.1 Using MQ Explorer

Using MQ Explorer

1. Expand the Queue Manager folder in the Navigator view in WebSphere MQ v7.0 explorer, and then right-click **Subscriptions**.

   There are two options on the context menu, New - Subscription and Status, as shown in Figure 9-19.

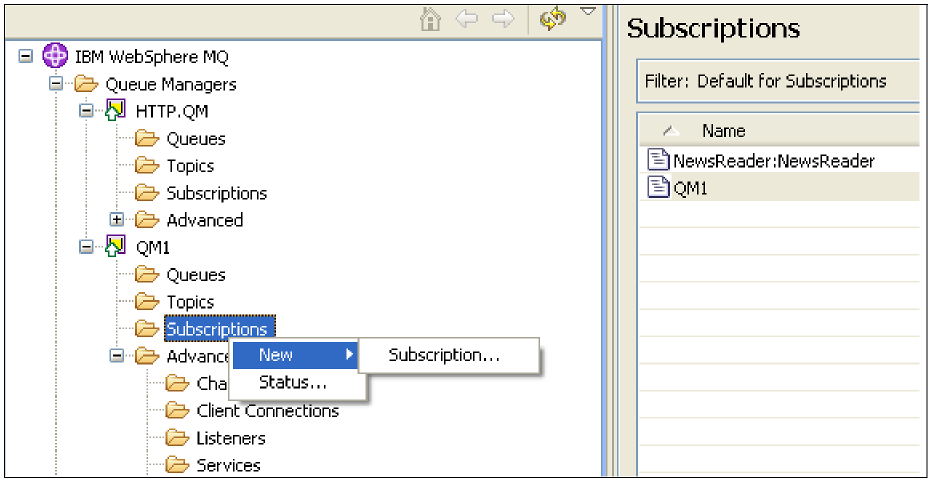
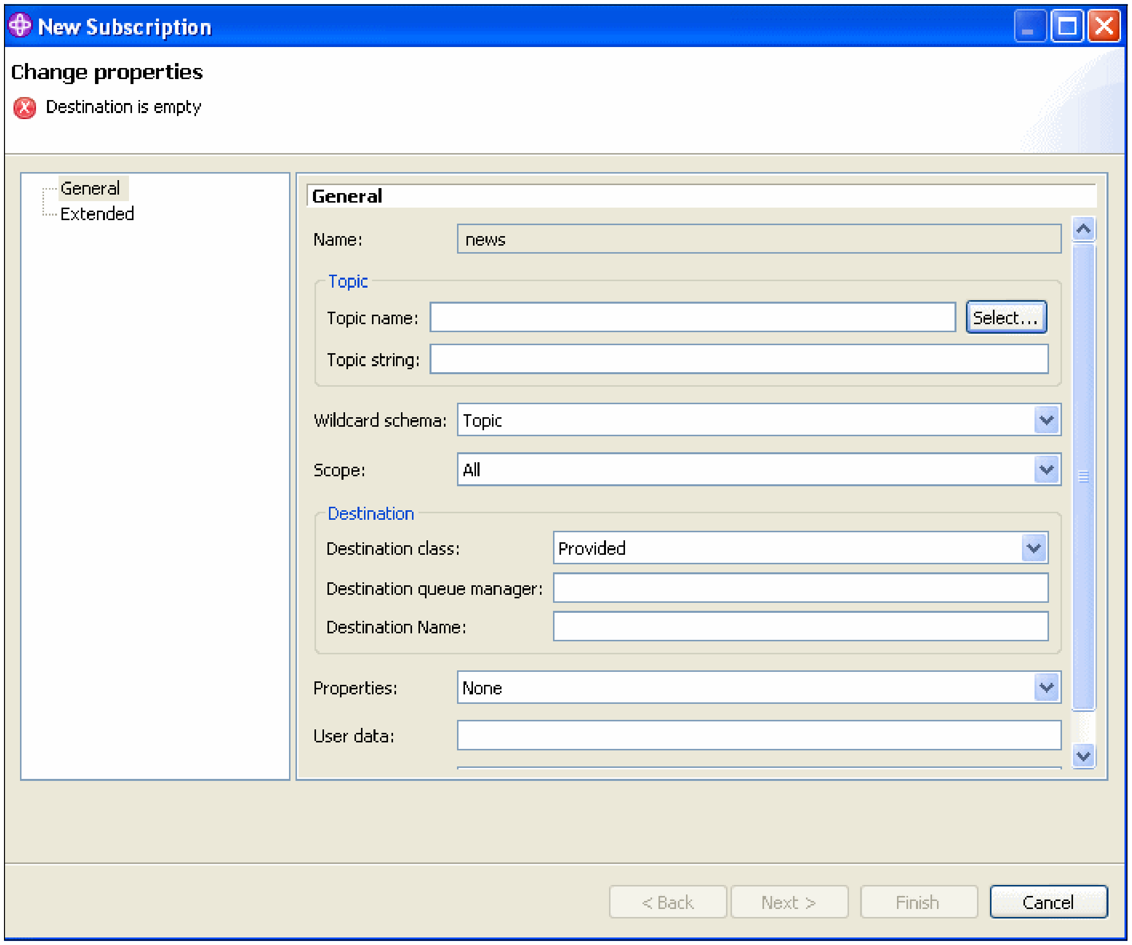


Figure 9-19 Subscriptions: Right-click menu

2. Click **Subscription** and the New Subscription window will be displayed, as shown in Figure 9-20. Enter a name for the subscription and then click **Next**.


New Subscription

Change properties

Destination is empty

General

Extended

General

Name: news

Topic

Topic name:

Topic string:

Wildcard schema: Topic

Scope: All

Destination

Destination class: Provided

Destination queue manager:

Destination Name:

Properties: None

User data:

Figure 9-20 Create subscription

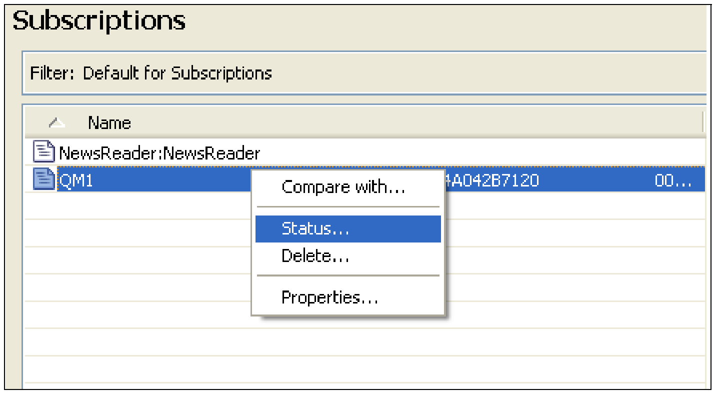
The topic name and the topic string properties refer to the topic object name and the corresponding string in the topic hierarchy, such as MATT.RETAIL.CAT and matt/retail/cat.

Wildcard characters can be used in the topic string to subscribe to multiple topics. Refer to 4.2.2, “Topic strings and topic objects” on page 45, for further details.

Scope refers to the subscription scope (SUBSCOPE), which can be set to either Queue Manager (the subscription forwards messages published on the topic only within the local queue manager) or ALL (the subscription is forwarded to all queue managers directly connected through a Publish/Subscribe collective or hierarchy).

Destination class can be set to managed (the queue manager internally manages the queuing of messages for the subscription) or provided (a non-managed subscription where the queue for this subscription is provided by the subscription definition). The destination queue name is entered in the destination property and the destination queue manager is entered in the destination queue manager property.

3. Right-click a subscription to display a context menu, as shown in Figure 9-21.

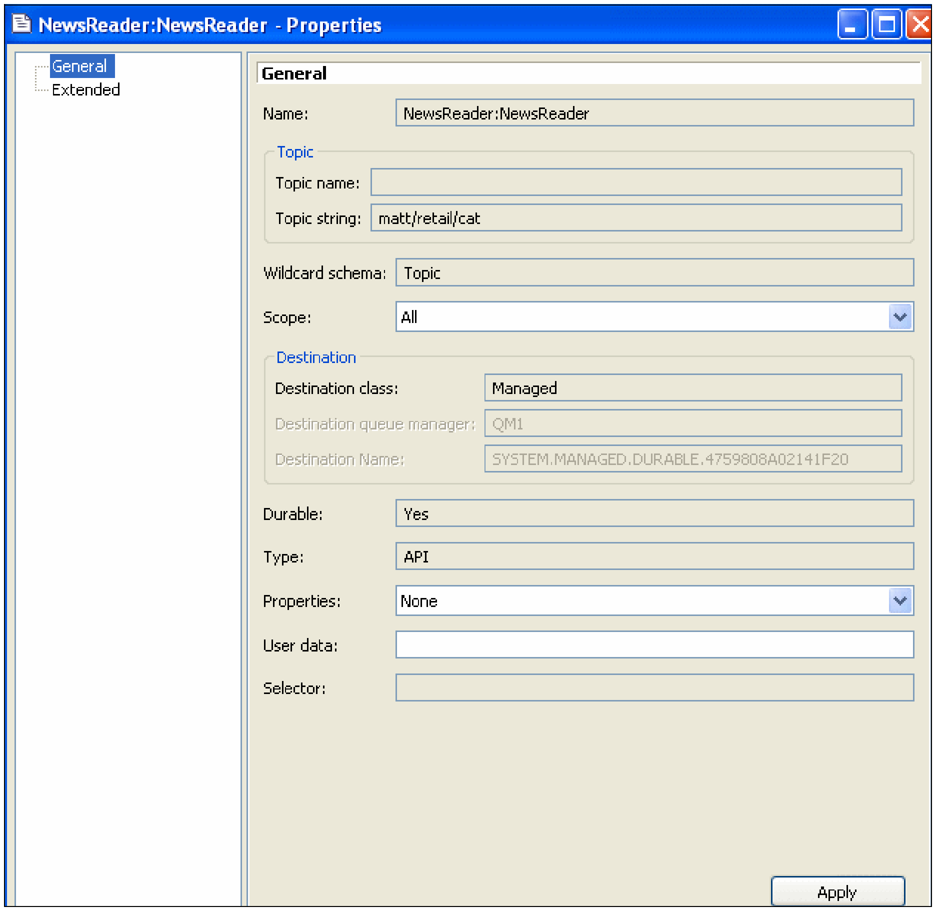

Subscriptions

Filter: Default for Subscriptions

Name		
NewsReader:NewsReader		
QM1	4A042B7120	00...

Figure 9-21 Subscription context menu

4. Clicking **Compare With** helps to compare this subscription with other subscriptions. Clicking **Status** displays the status of the subscription. Clicking **Properties** displays a window that allows the subscription properties to be changed, as shown in Figure 9-22.


NewsReader:NewsReader - Properties

General
Extended

General

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:	

Apply

Figure 9-22 Changing subscription properties

9.2.2 Using MQSC

WebSphere MQ v7.0 introduces new MQSC commands for managing subscriptions. The commands accept a variety of parameters that are described in the manual *Script (MQSC) Command Reference*, available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

Define subscription

The DEFINE SUB command defines a new administrative subscription. The subscribed topic is referred to by topic object (using the TOPICOBJ parameter) or topic string (using the TOPICSTR parameter).

The DESTCLAS parameter can be set to two values:

- MANAGED: This is a managed subscription. The queue manager internally manages the queuing of messages for the subscription.
- PROVIDED: This is a non-managed subscription where the queue for this subscription is provided by the subscription definition. The name of the queue and queue manager is specified by the DEST and DESTQMGR parameters, respectively. DESTCLAS defaults to PROVIDED if the parameter is not specified. DESTQMGR defaults to blank (the local queue manager) if it is not specified.

The subscription can participate in a distributed Publish/Subscribe environment by using the SUBSCOPE parameter. It can be assigned the following values:

- ALL: The subscription is forwarded to all queue managers directly connected through a Publish/Subscribe collective or hierarchy. This is the default if the SUBSCOPE parameter is not specified.
- QMGR: The subscription forwards messages published on the topic only within the local queue manager.

Example usage of this command is given in Example 9-3.

Example 9-3 Usage of the DEFINE SUB command

```
DEFINE SUB(SUB.RETAIL.CAT) TOPICOBJ(MATT.RETAIL.CAT) DESTCLAS(MANAGED)
DEFINE SUB(SUB.MATTRETCAT) TOPICSTR('matt/retail/cat')
   DESTCLAS(MANAGED)
DEFINE SUB(SUB.PROVCAT) TOPICSTR('matt/retail/cat')
   DEST(SUB.PROVCAT.DESTQ)
```

Display subscription

Details of existing administrative subscriptions can be displayed using the DISPLAY SUB command. The values of all parameters can be displayed or only those that have been specified. The display can also be limited by filter conditions, durability, and subscription type. See Example 9-4.

Example 9-4 Usage of the DISPLAY SUB command

```
DISPLAY SUB(SUB.PROVCAT) ALL
DIS SUB(*) DESTCLAS DEST DESTQMGR
```

Alter subscription

Existing subscriptions can be altered using the ALTER SUB command. See Example 9-5.

Example 9-5 Usage of the ALTER SUB command

```
ALTER SUB(SUB.PROVCAT) DEST(SUB.PROVCAT.DQ1) DESTCORL(1)
```

Delete subscription

Existing subscriptions can be deleted using the DELETE SUB command. This requires the unique hexadecimal subscription ID (SUBID) that was assigned by the queue manager when the subscription was created. The SUBID is displayed by the DISPLAY SUB command. See Example 9-6.

Example 9-6 Usage of the DELETE SUB command

```
DELETE SUB(SUB.MATTRETCAT)
   SUBID(414D51204D5147425431202020202020206D3F734820002503)
```

9.3 Displaying Publish/Subscribe status

The status of the Publish/Subscribe environment can be displayed using MQSC commands. Additional parameters for the commands are described in the manual *Script (MQSC) Command Reference*, available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

9.3.1 Display Pub/Sub status

The DISPLAY PUBSUB command can be used to display the Publish/Subscribe status information about a queue manager. The general form of the command and an example are given in Example 9-7.

Example 9-7 Usage of the DISPLAY PUBSUB command

```
DISPLAY PUBSUB TYPE(type) ALL|QMNAME|STATUS
DIS PUBSUB TYPE(ALL) ALL
```

The TYPE parameter allows a preference to be specified for the type of information to be displayed for a distributed Publish/Subscribe environment. It can take the following values:

- ALL: Displays the pub/sub status for this queue manager and for parent and child hierarchical connections
- CHILD: Displays the pub/sub status for child connections
- LOCAL: Displays the pub/sub status for this queue manager
- PARENT: Displays the pub/sub status for the parent connection

The displayed information includes the STATUS parameter, which may have the following values, depending on the type of information requested.

- If the TYPE entered was LOCAL, the following values can be returned:
  - ACTIVE: The pub/sub engine and the queued pub/sub interface are running. It is therefore possible to publish or subscribe using the APIs and to the queues that are monitored by the queued pub/sub interface.
  - COMPAT: The pub/sub engine is running. It is therefore possible to publish or subscribe using the APIs. The queued pub/sub interface is not running. Therefore, any message that is put to the queues that are monitored by the queued pub/sub interface is not acted upon by WebSphere MQ v7.0.
  - ERROR: The pub/sub engine has failed. Check your error logs to determine the reason for the failure.
  - INACTIVE: The pub/sub engine and the queued pub/sub interface are not running. It is therefore not possible to publish or subscribe using the APIs. Any pub/sub messages that are put to the queues that are monitored by the queued pub/sub interface are not acted upon by WebSphere MQ V7.0.
  - STARTING: The pub/sub engine is initializing and is not yet operational.
  - STOPPING: The pub/sub 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:
    - Transmit queue is not defined.
    - 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.
  - REFUSED: The connection has been refused by the parent queue manager.
  - STARTING: The queue manager is attempting to request that another queue manager becomes its parent.
  - STOPPING: The queue manager is disconnecting from its parent.

9.3.2 Display subscriber status

The DISPLAY SBSTATUS command can be used to display the status of a given subscription. The display can also be limited by filter conditions, durability, and subscription type. The general form of the command and two examples are given in Example 9-8.

Example 9-8 Usage of DISPLAY SBSTATUS command

```
DISPLAY SBSTATUS(subscription_name) SUBTYPE(USER|PROXY|ADMIN|API|ALL)
   DURABLE(ALL|NO|YES) ALL|status_attributes
DISPLAY SBSTATUS SUBID(subscription_id) ...

DIS SBSTATUS(*) ALL
DIS SBSTATUS(MATT*) DURABLE(YES) LMSGDATE LMSGTIME
```

10

WebSphere MQ Bridge for HTTP

This chapter describes the WebSphere MQ Bridge for HTTP that enables client applications to exchange messages with WebSphere MQ from any platform or language with HTTP capability.

This chapter contains the following sections:

- 10.1, “Overview” on page 226
- 10.2, “Prerequisites” on page 227
- 10.3, “Supported verbs” on page 228
- 10.4, “HTTP request and response” on page 229

10.1 Overview

WebSphere MQ Bridge for HTTP is a feature of WebSphere MQ V7.0 that allows client applications to interact with a queue manager using the HTTP protocol. Clients can perform several messaging functions with WebSphere MQ from any platform or language that has HTTP capability, without the need for WebSphere MQ client libraries on the platform.

A key feature of WebSphere MQ is its ubiquity. It is available on a wide range of platforms and operating systems. This feature adds to the fold any platform capable of issuing HTTP requests. Clients with zero footprint (without any WebSphere MQ libraries and software installed on the client side) can use this feature to perform messaging using WebSphere MQ.

Figure 10-1 shows that the WebSphere MQ Bridge for HTTP is hosted by an application server as a servlet application. It receives HTTP requests from one or more clients, interacts with WebSphere MQ on their behalf, and returns HTTP responses back to the clients.

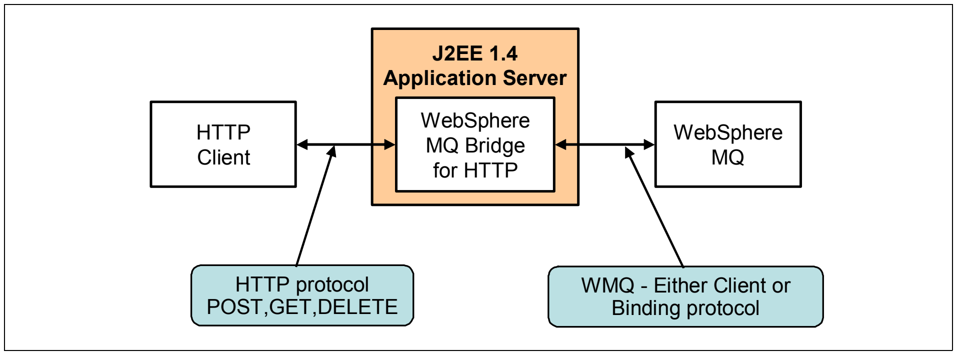

J2EE 1.4
Application Server
HTTP Client
WebSphere MQ Bridge for HTTP
WebSphere MQ
HTTP protocol POST,GET,DELETE
WMQ - Either Client or Binding protocol

Figure 10-1 WebSphere Bridge for HTTP

Features of WebSphere MQ Bridge for HTTP

The features are:

- Because of the low quality of service inherent in HTTP, the WebSphere MQ Bridge for HTTP is not suitable for use with messages where guaranteed delivery is required.
- HTTP supports both point-to-point and Publish/Subscribe messaging paradigms. The interface is subset of the full Publish/Subscribe functionality offered by WebSphere MQ.
- Using the HTTPS functionality of a supported application server, the bridge can used both HTTP and HTTPS protocols.

- It uses an application server to provide the HTTP server-side stack.
- The WebSphere MQ Bridge for HTTP consists of a servlet that is connected to a JCA resource adapter to WebSphere MQ in either client or binding mode.

For further information about the WebSphere MQ Bridge for HTTP, refer to:

http://www.ibm.com/software/integration/wmq/httpbridge/

Note: The WebSphere MQ Bridge for HTTP is also available for WebSphere MQ V6.0 as SupportPac MA0Y. For information about this SupportPac refer to:

http://www.ibm.com/support/docview.wss?uid=swg24016142

Note: There is an alternative method of accessing WebSphere MQ from a HTTP client. SupportPac MA94 provides a stand-alone native HTTP listener that does not require an application server. It serves HTML pages and supports both point-to-point and Publish/Subscribe messaging. However, it may not provide the same functions as SupportPac MA0Y or the WebSphere MQ V7.0 Bridge for HTTP. For further information refer to:

http://www.ibm.com/support/docview.wss?uid=swg24017593

10.2 Prerequisites

The following prerequisites apply to the WebSphere MQ Bridge for HTTP:

- WebSphere Application Server Version 6.0.2.1 and later or WebSphere Application Server Community Edition Version 1.1 or later. The WebSphere MQ Bridge for HTTP may be used with other J2EE™ 1.4 compliant application servers, but it is not supported by IBM.
- A WebSphere MQ JMS provider within the application server.
- To use WebSphere Application Serve Version 6 or earlier, WebSphere Application Serve Message Listener Port (MLP) is required to integrate WebSphere MQ as the JMS provider.
- With an application server other than WebSphere Application Server, use the WebSphere MQ resource adapter. This is included in WebSphere MQ V7.0.
- The WebSphere MQ Bridge for HTTP is supplied as a .war file. It must be deployed to the application server as a servlet application.

10.3 Supported verbs

The bridge is capable of handling three types of HTTP request:

- POST
- GET
- DELETE

The WebSphere MQ function of each HTTP verb is listed in Table 10-1.

Table 10-1 WebSphere MQ Bridge for HTTP verbs

HTTP request	WebSphere MQ function
POST	Puts a message on a queue or topic.
GET	Browses the first message on a queue. In line with the HTTP protocol, this does not delete the message from the queue. This verb cannot be used with Publish/Subscribe messaging.
DELETE	Browses and deletes a message from a queue or topic.

The bridge runs as a servlet on the application server. It is limited by the restrictions of HTTP protocol, so the quality of service is 0 (in other words, fire-and-forget with no guarantee of delivery).

The client issues an HTTP request to the application server hosting the WebSphere MQ/HTTP bridge. The URL and payload of this request determines the message data, the destination, and other options. The WebSphere MQ queue manager sees the bridge as an ordinary remote client application or local binding application.

A typical flow to put a message on a queue is:

1. The client makes a HTTP POST request to a URL on the application server.
2. The bridge interprets the request and presents it as a JMS operation to the WebSphere MQ queue manager, which then performs the messaging request.
3. The bridge receives the status code and associated data back from JMS, such as the message ID of the message put by the queue manager.
4. The bridge code returns an HTTP response to the client indicating the success or failure of the operation. The response can contain headers with the requested message properties.

Note: The client can be anything that can invoke an HTTP request and handle an HTTP response (for example, Web browser, perl, python, ruby).

The bridge uses the representational state transfer (REST) programming model, so no record is maintained of messages delivered to the client. Subsequent requests to the same topic may result in duplicate messages being received by an HTTP client. Multiple retained messages on sub-topics cannot be consumed. For example, if topic “weather/NewYork” contains a retained message and topic “weather/London” contains a retained message, a request to topic “/weather/#” will return the first message to the client, not both.

10.4 HTTP request and response

The purpose of the WebSphere MQ bridge for HTTP is to receive HTTP requests from clients, interact with WebSphere MQ, and return HTTP responses to the client. This section deals with the format of messages, constructing HTTP requests and handling HTTP responses.

Table 10-2 shows the mapping of POST, GET, and DELETE verbs to WebSphere MQ API calls for messages and Publish/Subscribe topics.

Table 10-2 WebSphere MQ HTTP verbs

		HTTP verb mapping			
Resource	**Sample URIs**	**GET**	**POST**	**PUT**	**DELETE**
Messages	http://host/msg/queue/qname/	MQGET	MQPUT	-	MQGET
Topic	http://host/msg/topic/tname/	-	MQPUT	-	MQGET

To put a message to a queue, the client creates an HTTP request. This can be done using a HTTP programming method, such as AJAX or Java HTTP libraries.

All Message Descriptor fields are conveyed in HTTP headers prefixed with 'x-msg-'. The message body is passed in the HTTP entity body, and message type in the HTTP content type.

The following sub-sections provide more information about the URI format and HTTP verbs.

10.4.1 URI FORMAT

The URI format supported by WebSphere MQ Bridge for HTTP is shown in Example 10-1.

Example 10-1 URI format in WebSphere MQ Bridge for HTTP

```
Wmq-http-iri = "http:" "//" connection-name "/" wmq-dest
 connection-name = [ tcp-connection-name ]
 tcp-connection-name = ihost [ ":" port ]
 wmq-dest = queue-dest | topic-dest
 queue-dest = "msg/queue/" wmq-resource [ "@" wmq-qmgr ] "/"
 topic-dest = "msg/topic/" wmq-resource "/"
```

The context root is defined in the deployment plan when the bridge servlet is deployed into the application server. The bridge is configured such that all requests to URIs for message queues and message topics are handled by the bridge.

10.4.2 HTTP POST

HTTP POST can be used to put a message onto a WebSphere MQ queue or topic. The WebSphere MQ Bridge for HTTP will put messages to queues and topics as either WebSphere MQ messages (without a MQRFH2 header) or JMS messages (with a MQRFH2 header). This is dependent on the content-type of the request, and the value of the x-msg-class header in the request.

Use HTTP headers in the HTTP POST request to set the properties of the message that is put to WebSphere MQ. The require-headers header is used to specify the information about the WebSphere MQ message to be received as headers in the response message.

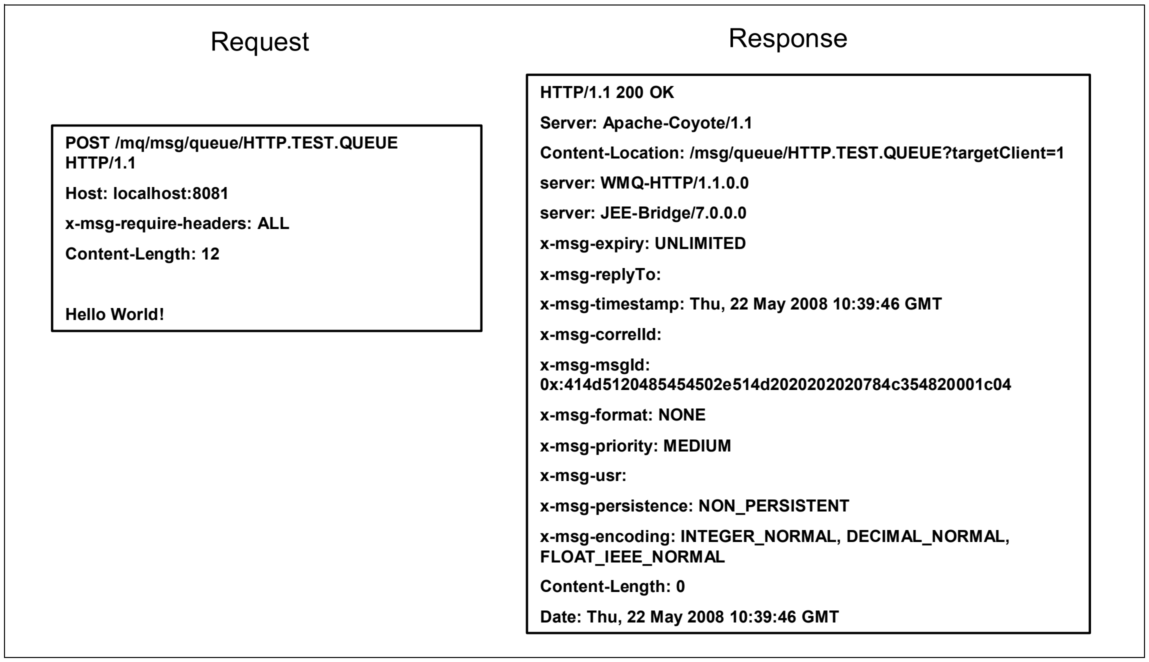
If the HTTP POST request is successful, the entity of the response message will be empty and the content-length of the response will be set to zero. The status code of the HTTP response will be '200 OK'.

In the case of an unsuccessful request, the entity of the response will contain a WebSphere MQ Bridge for HTTP error message and the status code.

The status code of the HTTP response will be set to one of the values listed in the section WebSphere MQ Bridge for HTTP → HTTP Return codes of the *Web Services* manual, available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

Figure 10-2 details a sample HTTP POST flow.


Request
Response

```
POST /mq/msg/queue/HTTP.TEST.QUEUE
HTTP/1.1
Host: localhost:8081
x-msg-require-headers: ALL
Content-Length: 12

Hello World!
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Location: /msg/queue/HTTP.TEST.QUEUE?targetClient=1
server: WMQ-HTTP/1.1.0.0
server: JEE-Bridge/7.0.0.0
x-msg-expiry: UNLIMITED
x-msg-replyTo:
x-msg-timestamp: Thu, 22 May 2008 10:39:46 GMT
x-msg-correlId:
x-msg-msgId:
0x:414d5120485454502e514d2020202020784c354820001c04
x-msg-format: NONE
x-msg-priority: MEDIUM
x-msg-usr:
x-msg-persistence: NON_PERSISTENT
x-msg-encoding: INTEGER_NORMAL, DECIMAL_NORMAL,
FLOAT_IEEE_NORMAL
Content-Length: 0
Date: Thu, 22 May 2008 10:39:46 GMT
```

Figure 10-2 Sample HTTP flow: POST

10.4.3 HTTP GET

HTTP GET can be used to browse a message from a WebSphere MQ queue. HTTP GET is not supported for use with topics. The entity of the response message that is sent back to the client contains the data from the WebSphere MQ message. The WebSphere MQ message remains on the queue.

HTTP headers can be used in a HTTP GET request as follows:

1. The x-msg header specifies the information about the WebSphere MQ message to be received as headers in the response message.
2. The correlID header or msgID header, or both, select the message to be browsed from the queue.
3. The wait header determines how long to wait for a message to arrive on the queue.

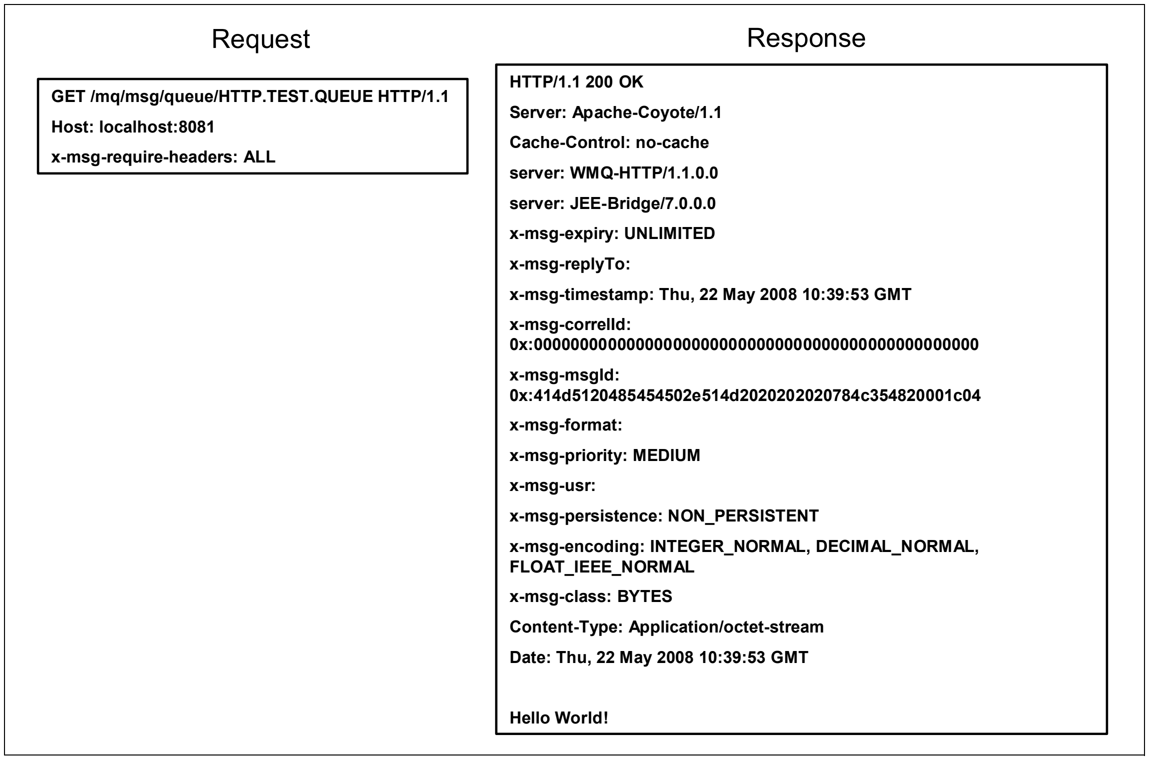
If the HTTP GET request is successful, the entity of the response message will contain the message retrieved from the WebSphere MQ queue and the HTTP content-length headers will be set to the number of bytes in the entity body. The status code of the HTTP response will be '200 OK'.

In the case of an unsuccessful request, the response will contain a WebSphere MQ Bridge for HTTP error message and the status code.

The status code of the HTTP response will be set to one of the values listed in the section WebSphere MQ Bridge for HTTP → HTTP Return codes of the *Web Services* manual, available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

Figure 10-3 details a sample HTTP GET flow.



Request
Response

```
GET /mq/msg/queue/HTTP.TEST.QUEUE HTTP/1.1
Host: localhost:8081
x-msg-require-headers: ALL
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache
server: WMQ-HTTP/1.1.0.0
server: JEE-Bridge/7.0.0.0
x-msg-expiry: UNLIMITED
x-msg-replyTo:
x-msg-timestamp: Thu, 22 May 2008 10:39:53 GMT
x-msg-correlId:
0x:000000000000000000000000000000000000000000000000
x-msg-msgId:
0x:414d5120485454502e514d2020202020784c354820001c04
x-msg-format:
x-msg-priority: MEDIUM
x-msg-usr:
x-msg-persistence: NON_PERSISTENT
x-msg-encoding: INTEGER_NORMAL, DECIMAL_NORMAL,
FLOAT_IEEE_NORMAL
x-msg-class: BYTES
Content-Type: Application/octet-stream
Date: Thu, 22 May 2008 10:39:53 GMT

Hello World!
```

Figure 10-3 Sample HTTP flow: GET

10.4.4 HTTP DELETE

The HTTP DELETE operation can be used to get a message from a WebSphere MQ queue or topic. The message will be destructively removed from the queue or topic. A response message will be sent back to the client, including information about the message.

HTTP headers can be used in a HTTP DELETE request as follows:

1. The x-msg header specifies the information about the WebSphere MQ message to be received as headers in the response message.
2. The correlID header or msgID header, or both, determine the message to be got from the queue or topic.
3. The wait header determines how long to wait for a message to arrive on the queue or topic.

If the HTTP DELETE request is successful, the entity of the response message will contain the data of the message retrieved from the WebSphere MQ queue or topic and the HTTP content-length headers will be set to the number of bytes in the entity body.

In case of an unsuccessful request, the response will contain a WebSphere MQ Bridge for HTTP error message and the status code.

The status code of the HTTP response will be set to one of the values listed in the section WebSphere MQ Bridge for HTTP → HTTP Return codes of the *Web Services* manual, available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

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.

This chapter contains the following sections, which discuss each of the enhancements:

For additional information refer to the following WebSphere MQ manuals:

- *WebSphere MQ System Administration Guide V7.0*
- *WebSphere MQ z/OS Concepts and Planning Guide V7.0*
- *WebSphere MQ z/OS Systems Setup Guide V7.0*
- WebSphere MQ Publish/Subscribe Users Guide V7.0
- *WebSphere MQ Security V7.0*

These manuals are available in the IBM Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

11.1 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 43.

11.2 RACF mixed case classes and profiles

WebSphere MQ V7.0 supports the use of mixed case RACF profiles. Five new RACF classes have been added and a new queue manager parameter introduced.

11.2.1 New queue manager parameter SCYCASE

The new queue manager parameter SCYCASE (security case) can take the value UPPER or MIXED to determine which RACF profiles are used by the queue manager to provide security:

- UPPER: Only uppercase profiles are used, as per the behavior of WebSphere MQ V6.0.
- MIXED: Only mixed-case profiles are used.

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 MQSC command DISPLAY QMGR SCYCASE displays the current setting of the SCYCASE parameter for the queue manager. This parameter is only valid on z/OS.

MQ Explorer displays the current setting of the SCYCASE parameter for z/OS queue managers as *Security profile case* in the Extended section of the Queue Manager Properties window.

11.2.2 New RACF classes

Five new RACF classes are added in WebSphere MQ V7.0. They allow profiles to be defined to protect mixed-case object names and administration functions. Four of these new classes replace equivalent classes that allowed uppercase-only profiles.

Table 11-1 New RACF classes

New (mixed case) RACF class name	**Previous (upper case only) RACF class name**	**Use**
MXADMIN	MQADMIN	Administration
MXQUEUE	MQQUEUE	Queues
MXPROC	MQPROC	Processes
MXNLIST	MQNLIST	Name lists
MXTOPIC	None	Publish/Subscribe topics

The new MXTOPIC class does not have an equivalent uppercase class.

Previously, mixed-case object names could only be protected by using generic uppercase profiles in the appropriate uppercase RACF class. Mixed-case objects can now be protected by using mixed-case profiles (generic or specific) in the appropriate mixed-case RACF class.

11.2.3 Using mixed case profiles

With only uppercase RACF profiles available, the only way to protect resources with mixed-case names is to use generic profiles. For example, the following RACF command would define a profile applying to all queue names that began QMHQ.PAYROLL., including those with mixed-case names:

```
RDEFINE MQQUEUE QMHQ.PAYROLL.**
```

Therefore, QMHQ.PAYROLL.inqueue and QMHQ.PAYROLL.outqueue would be protected by the same profile.

The mixed-case class MXQUEUE can be used to define profiles to provide explicit protection for queues with mixed-case names, for example:

```
RDEFINE MXQUEUE QMHQ.PAYROLL.inqueue
RDEFINE MXQUEUE QMHQ.PAYROLL.outqueue
```

Different access restrictions can now be granted to these queues.

11.2.4 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 that 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.

11.2.5 Migrating to mixed-case security

To use mixed-case security profile support:

1. Ensure that the new WebSphere MQ RACF classes are installed and active.
2. Copy all your existing profiles and access levels from the uppercase classes to the equivalent mixed case class:
   a. MQADMIN to MXADMIN
   b. MQPROC to MXPROC
   c. MXLIST to MXNLIST
   d. MQQUEUE to MXQUEUE
3. Start the queue manager subsystem with the queue manager SCYCASE attribute set to UPPER.
4. Change the value of the SCYCASE attribute to MIXED by issuing the following command:
   ```
   ALTER QMGR SCYCASE(MIXED)
   ```
5. Activate the security profiles by issuing the following command:
   ```
   REFRESH SECURITY(*) TYPE(CLASSES)
   ```
6. Test that security profiles are working correctly.

11.3 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 that 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 that 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 11-1 The following MQSC command specifies a limit of five 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 )
```

We recommend that security should be imposed on the SYSTEM.ADMIN.SVRCONN channel by using Secure Sockets Layer (SSL) or a security exit to restrict access to authorized administrators. Security enhancements for remote administration using MQ Explorer are covered in 8.1.5, “Security” on page 167. This contains all the steps required to set up security on the Explorer side.

11.4 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 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 then opens.
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.

11.5 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 that is normally used.

To run an Open TCB, code must be thread-safe. This means that 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) must 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.

12

Installation and migration

This chapter provides guidance on some of the considerations that 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.

It is also not possible to fully address migration of Publish/Subscribe from the queued interface provided in WebSphere MQ V6.0 or from WebSphere Message Broker. The appropriate safe migration path depends on how these facilities are being used and the complexity of the environment. Migration will almost certainly involve application code changes and environment configuration changes. A deep understanding of the migration issues is required and detailed planning must be carried out to ensure a successful migration. WebSphere MQ Publish/Subscribe migration is therefore beyond the scope of this book.

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.

Many references are made in this chapter to WebSphere MQ manuals that provide more information. These are available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

This chapter contains the following sections:

12.1 Hardware and software prerequisites

The hardware and software prerequisites for WebSphere MQ V7.0 are described in the relevant *WebSphere MQ Quick Beginnings* manual for the distributed platforms, and in the *WebSphere MQ for z/OS System Setup Guide* and *WebSphere MQ for z/OS Program Directory* for the z/OS platform.

Detailed migration information is found in *WebSphere MQ V7.0 Migration Information.* We strongly recommend reading this manual.

National Language Support on AIX

WebSphere MQ V7.0 requires an AIX® operating system at level 5.3 or later. 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.

12.2 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* manual for the distributed platforms, and for z/OS in the *WebSphere MQ for z/OS System Setup Guide*, *WebSphere MQ for z/OS Concepts and Planning Guide*, and the *WebSphere MQ for z/OS Program Directory*.

12.3 Co-existence with previous versions

On distributed platforms it is not possible to install and run WebSphere MQ V7.0 to coexist with an earlier version of WebSphere MQ on the same server. Installing WebSphere MQ V7.0 always upgrades any earlier version of WebSphere MQ that has been previously installed on the server.

Coexistence 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 that the current environment is stable with the PTFs before proceeding to the upgrade.

WebSphere MQ V7.0 for z/OS does not support queue-sharing groups (QSGs) shared with subsystems (queue managers) running at previous versions of WebSphere MQ. All subsystems within a queue-sharing group must be migrated to V7.0 at the same time, or the feature must be disabled until all subsystems are running at that version.

WebSphere MQ V7.0 interoperates within a network of existing queue managers running on all supported platforms with previous versions of WebSphere MQ, using clients, distributed queuing, and MQ Clusters.

For managing coexistence of message properties between MQ V7.0 and other versions, refer to the PROPCTL attribute on channels and queues, as described in the relevant *WebSphere MQ Quick Beginnings* manual for distributed platforms.

12.4 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 general guidance on actions that can be taken to ensure a smooth migration.

Prior to the installation or migration of any software we recommend that a full system backup is performed to ensure that the system can be reverted to its original working environment in the event that unresolved problems are encountered during the migration.

12.4.1 General migration considerations

General issues to consider when planning a migration are:

- Detailed migration information is provided in the manual *WebSphere MQ V7.0 Migration Information*.
- Consult the *WebSphere MQ Quick Beginnings Guide* for specific distributed platforms and the *WebSphere MQ z/OS System Setup Guide*.
- Review the latest readme files that are shipped with the product. They may contain information that 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 that 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 that 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:
  a. Copy all resources from the server connected to another server.
  b. Perform a migration on the duplicate server.
  c. 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 domain name system (DNS) entries.

12.4.2 Queue manager migration

Before migrating to WebSphere MQ V7.0, ensure that all prerequisites are satisfied and that the current environment is backed up.

Ensure that applications are thoroughly tested against the product before deploying into production. We recommend 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.

12.4.3 Migration steps

The following is an example of steps that 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 that connect to this queue manager.
2. Stop the listeners.
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. Save or 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. Shut down the WebSphere MQ queue managers and associated services, such as dead letter queue handlers and trigger monitors.
9. Back up 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 that 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 managers and wait for them 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 start the MQ TCP/IP port listener, start the listener process `runmqlsr` or enable the port in inetd.
14. Start channels that were stopped before the upgrade.

Details of tools and commands to perform these procedures on various platforms are beyond the scope of this book.

12.4.4 Fallback considerations

In the event that the migration must be backed out to the previous version, we recommend the following steps. 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 that connect to this queue manager.
2. Stop the listeners.
3. Stop all applications that can connect to the queue manager.
4. Stop system management products.
5. Save or unload all messages currently queued to a file.

Note: Saved messages may not be usable in previous versions of WebSphere MQ due to differences in header formats and properties.

6. Save all queue manager attributes and object definitions.
7. Shut down the WebSphere MQ queue managers 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 definitions 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 pagesets.

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.

Note: Not all messages saved from WebSphere MQ V7.0 can be safely loaded into a previous version of WebSphere MQ due to differences in header formats and properties. This particularly applies to messages on SYSTEM queues, messages that use properties, and messages for Publish/Subscribe.

13. Perform verification tests.
14. Start the listener and channels and so on.
15. Continue to monitor the system health to ensure that the prior version of WebSphere MQ is functioning correctly.

12.4.5 Publish/Subscribe engine

WebSphere MQ V7.0 provides a new Pub/Sub engine that is integrated into the queue manager. Pub/Sub was previously available with WebSphere MQ V6.0 (and WebSphere MQ V5.3 with at least fix pack 8) on distributed platforms using a separate broker process that was associated with each queue manager. This Pub/Sub broker is now deprecated. However, applications written for it can work with WebSphere MQ V7.0 without modification.

To allow compatibility with the older Pub/Sub broker 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. There are also control commands to manage the migration of the old broker and its associated state.

Refer to the section “Migration to distributed Publish/Subscribe” in the *WebSphere MQ V7.0 Migration Information* manual and the section “Migrating to Version 7.0 Publish/Subscribe” in the *WebSphere MQ V7.0 Publish/Subscribe User’s Guide* for further details.

IBM encourages customers to migrate WebSphere MQ V6.0 Pub/Sub applications, objects, and configurations to use the new WebSphere MQ V7.0 Pub/Sub engine as soon as practical.

12.4.6 Java application migration considerations

There are considerations for the WebSphere MQ messaging provider in applications that use the WebSphere MQ classes for Java or Java Message Service. The WebSphere MQ messaging provider has two modes of operation:

- Normal mode is optimized to use the new features of WebSphere MQ V7.0.
- Migration mode is based on WebSphere MQ V6.0 function, and only those features that were available at this level can be used.

For further details refer to the section “Introduction to WebSphere MQ classes for JMS → What is new in WebSphere MQ Version 7.0? → WebSphere MQ Messaging Provider” in the *Using Java* manual.

12.4.7 General application migration considerations

For applications that do not use Publish/Subscribe, there are no migration processes or changes required for migration or installation of WebSphere MQ V7.0. Existing applications continue to function in their current state.

12.4.8 WebSphere MQ clients

Client channel definitions that 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 files are not migrated when upgrading to WebSphere MQ V7.0. Nor is the entire file updated when updating just one channel within the CCDT.

Client channels in CCDT files should be defined and maintained on a queue manager that is at the same version or earlier than the versions of all the clients that use the file.

As an alternative, IBM SupportPac MO72 MQSC Client for WebSphere MQ 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 a complete scenario that demonstrates how the new features and enhancements work and how to use them. The sample programs and scripts used for the scenario are available as text in Appendix A, “Scenario preparation scripts” on page 373, and for download by following the instructions in Appendix B, “Additional material” on page 379. This part consists of:

- Chapter 13, “Scenario overview” on page 255
- Chapter 14, “Scenario preparation” on page 269
- Chapter 15, “Scenario: Supplier pricing using Pub/Sub” on page 287
- Chapter 16, “Scenario: Store ordering with JMS” on page 303
- Chapter 17, “Scenario: News using client” on page 325
- Chapter 18, “Scenario: Web ordering over HTTP” on page 343
- Chapter 19, “Scenario: Warehousing using call back” on page 361
- Appendix A, “Scenario preparation scripts” on page 373
- Appendix B, “Additional material” on page 379

13 Scenario overview

Matt’s Deli is a chain of grocery stores. It is 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 be a guide to their best usage or to implement complete business logic.

This chapter contains the following sections:

13.1 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 feadquarters and they are delivered daily from the warehouse.

The newest venture is a Web deli where customers can order products directly on the Internet for home delivery. Headquarters negotiates prices from suppliers and arranges delivery of stock 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 is 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.

13.1.1 Business flow

The business flow is illustrated in Figure 13-1 on page 257. 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: A Web browser is used to place direct orders via the Web deli for products that are delivered from the warehouse. Internet customers can also receive news via an RSS feed about 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 that they wish to supply. Small suppliers may quote for only one or two items in the catalog, while large suppliers for many or all of them. These quotes are used by headquarters to choose the best suppliers 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 the 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 whether 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.

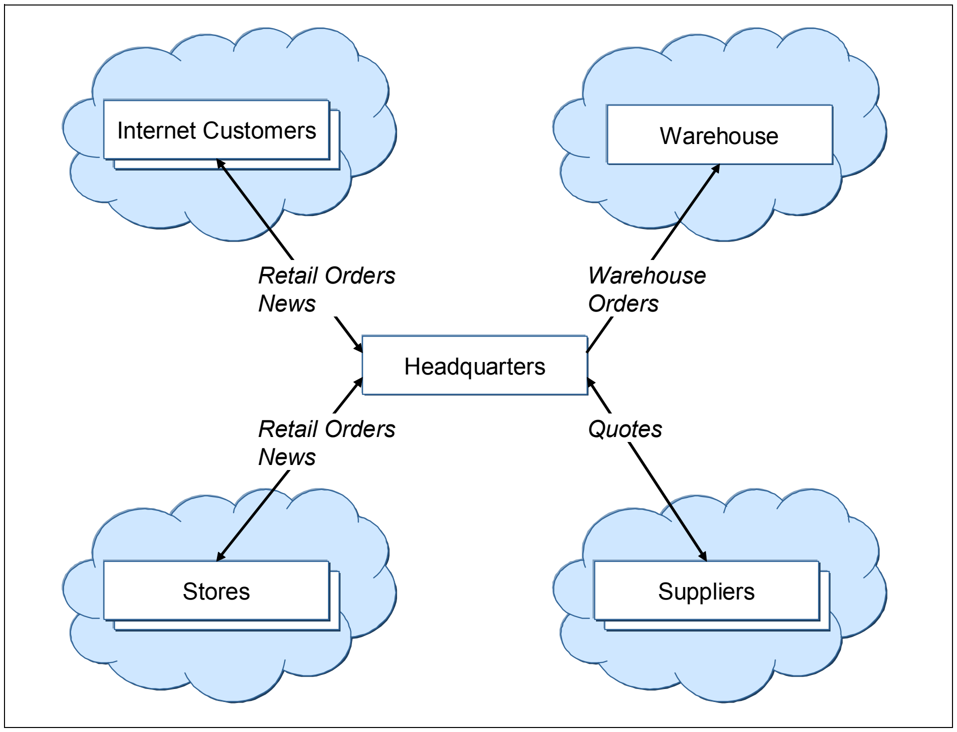


Figure 13-1 Matt’s Deli high-level data flow

13.1.2 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 must 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 that may have patterns of availability that differ 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.

13.1.3 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 extensive message parsing. The chosen payload format for all types of message is comma-separated values (CSV), where each field is separated from the next by a comma. All fields in all messages are simple text fields, allowing easy creation of test data.

One important field that 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.

13.2 Scenario implementation

Many of the new features in WebSphere MQ V7.0 are illustrated in this 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 that 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 WebSphere MQ Bridge for HTTP is used for Web ordering, showing how messages may send and receive 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 callback feature might be used. It also uses an administrative subscription to a remote queue.
- The retail price catalog illustrates use of retained publications.

13.2.1 Application flow

The scenario application flow is illustrated in Figure 13-2, 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.

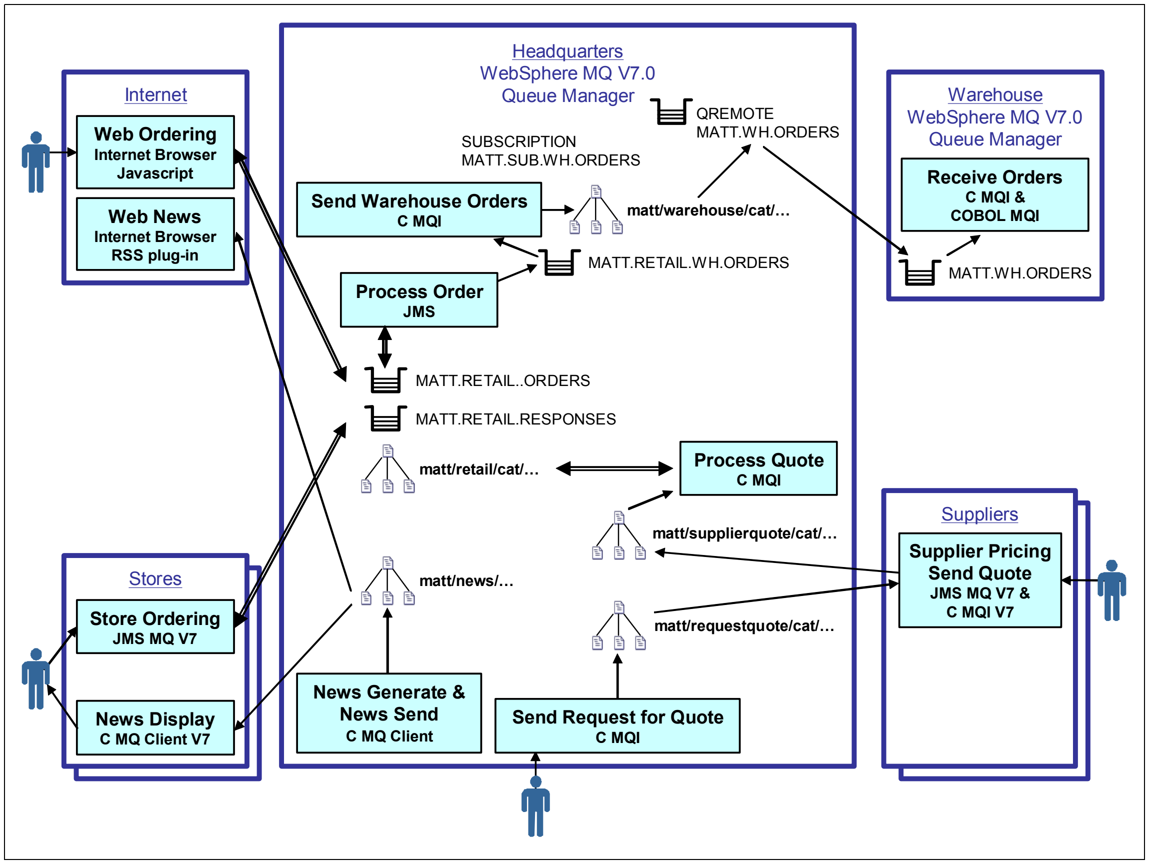


Figure 13-2 Matt's Deli application flows

13.2.2 Infrastructure

The scenario uses the following topic objects and queue objects that are defined on the headquarters queue manager.

Table 13-1 describes the topic object and strings used in this scenario.

Table 13-1 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.

Table 13-2 describes the queue objects used in the scenario.

Table 13-2 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.

13.3 Components

This scenario is divided into a number of distinct components that are covered in the forthcoming chapters. Most components can be set up and run independently or they can all be combined together into the complete scenario.

Complete source code, executable programs, and scripts are supplied in compressed files that are downloadable from the Internet. Refer to Appendix B, "Additional material" on page 379, and the scenario chapters for details on obtaining and using these files.

Note: The programs have been written to demonstrate the new and enhanced features of WebSphere MQ V7.0. The source code should not be relied on to use best practices for general programming or application design with WebSphere MQ. Therefore, the code should not be used as a model for developing real applications without a good understanding of the design methods used and their impact on performance and efficiency.

The remainder of this chapter provides a short overview of the purpose and flow of each component of our scenario.

13.3.1 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 15, "Scenario: Supplier pricing using Pub/Sub" on page 287.

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 that 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 that has its root at matt/requestquote/cat.

An illustrative flow for this component is as follows.

Matt’s Deli headquarters has a durable subscription to matt/supplierquote/cat# to ensure that it receives all quotes from suppliers.

Fred’s Fruit Farm is a supplier that supplies only fruit to Matt’s Deli and subscribes to the topic string matt/requestquote/cat/fresh/fruit/#.

Sam’s 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/#.

Mike’s Meat Market is a supplier that supplies only meat to Matt’s Deli and subscribes to matt/requestquote/cat/fresh/meat/#.

1. Matt’s Deli needs a price for red apples and so publishes a request quote message to matt/requestquote/cat/fresh/fruit/apple/red.
2. The message is made available to Fred’s Fruit Farm and Sam’s Super Supplier but *not* to Mike’s Meat market.
3. Sam’s Super Supplier chooses to provide a quote and sends a price quote message for a price of ten pounds to matt/supplierquote/cat/fresh/fruit/apple/red.
4. Matt’s Deli receives the message. As it is cheaper than our current price, Sam’s Super Supplier becomes our preferred supplier and we publish a modified retail price.
5. Fred’s Fruit Farm chooses to quote for red apples, too, and sends a quote message priced at nine pounds.
6. Matt’s Deli receives the message. As it is cheaper than our current price, Fred’s Fruit Farm becomes our preferred supplier and we publish a modified retail price.

13.3.2 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 plus sign (+) 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 16, “Scenario: Store ordering with JMS” on page 303.

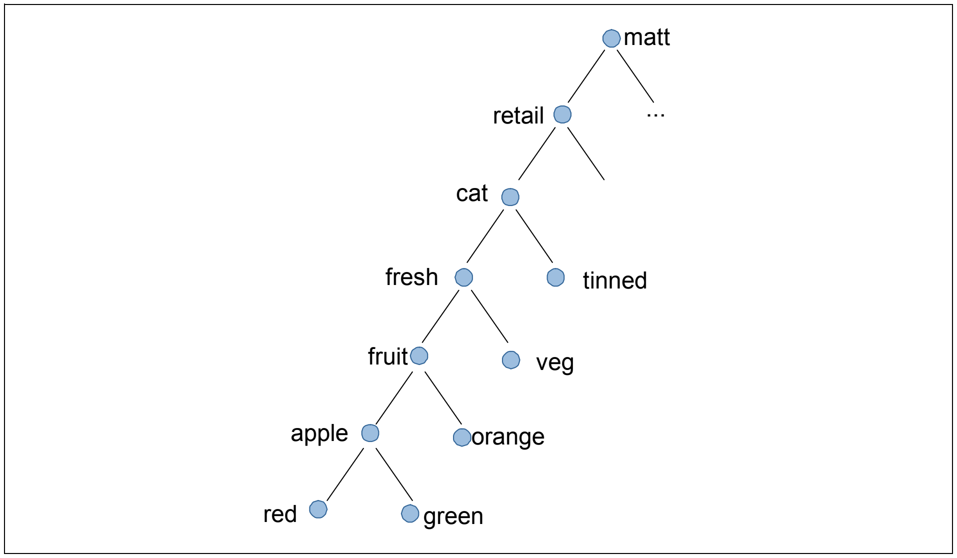


Figure 13-3 Sample fragment of retail catalog

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 that contains pricing information. The store ordering application navigates up and down the tree by using the plus sign (+) (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.

An illustrative flow for this component is as follows, making use of the sample retail catalog in Figure 13-3:

1. Store A of Matt’s Deli must order more red apples so that they start the store ordering application.
2. 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/+.
3. 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.
4. Selecting fresh causes the application to subscribe to topic string matt/retail/cat/fresh/+, this time returning fruit and veg.

5. Selecting fruit causes the application to subscribe to topic string matt/retail/cat/fresh/fruit/+, this time returning apple and orange.
6. 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.
7. Selecting red allows a quantity to be entered for an order. The application sends a retail order message to the MATT.RETAIL.ORDERS queue for processing at headquarters. It waits for a response message on the MATT.RETAIL.RESPONSES queue and displays the status of the order.

13.3.3 News

This component uses Publish/Subscribe to publish news items to multiple topics and demonstrates subscription to specific points in the topic tree. It uses security via topic objects to limit who can subscribe to the various news topics, as some of them may contain internal business information.

The programs use WebSphere MQ Client and WebSphere MQ classes for JMS to take advantage of two features, asynchronous put and read ahead, to provide improved performance when processing bulk distribution of news items.

Full details of the design and the operation of this component can be found in Chapter 17, “Scenario: News using client” on page 325.

An illustrative flow for this component is:

1. Matt’s Deli headquarters publishes news items to various topics under matt/news to ensure that the stores and public customers can receive relevant and up-to-date information about products and the operation of stores. There is also a JMS program that runs at headquarters that subscribes to public news topics and generates a RSS XML file.
2. Matt’s Deli stores subscribe to relevant news items during business hours and displays them on an overhead monitor for viewing by staff.
3. The public can subscribe to general news by running a WebSphere MQ Client program that displays all the news items as they are published. They can also view the most recent general news using a Web browser with an RSS plug-in or using a RSS reader program.

13.3.4 Web ordering

This component uses the WebSphere MQ Bridge for HTTP Bridge to illustrate how a Web browser client with no MQ code installed can access MQ. The Web page and the bridge are hosted by WebSphere Application Server. The bridge connects to a local WebSphere MQ queue manager via JMS to access the store ordering back-end application that is running at headquarters.

A JavaScript™ Web browser application uses a simple point-to-point interface to send a product order message to headquarters by performing an HTTP POST request. It then receives the order confirmation message by performing a HTTP DELETE request. It uses the same retail product order message protocol and queues as the store ordering component described in 13.3.2, “Store ordering” on page 263.

Full details of the design and the operation of this component can be found in Chapter 18, “Scenario: Web ordering over HTTP” on page 343.

An illustrative flow for this component is:

1. A customer opens the Matt’s Deli Web page using a Web browser.
2. On the displayed page, they complete the fields for product ID, quantity, and customer ID, and click a button to submit the order.
3. The Web page refreshes to show the status of the order. Another order can then be entered.

13.3.5 Warehousing

This component uses the MQI to illustrate the new callback 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 19, “Scenario: Warehousing using call back” on page 361.

An illustrative flow for this component is:

1. 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.
2. 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.

3. 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.
4. The red apple message is now transmitted to the warehouse systems queue manager.
5. 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.

14

Scenario preparation

This chapter describes the common scenario preparation that must be performed before running any particular scenario components from the scenario chapters in the rest of Part 3, “Scenario” on page 253.

This chapter contains the following sections:

- 14.1, “Environment setup” on page 270
- 14.2, “WebSphere MQ objects setup” on page 278

Any other setup and configuration tasks that are specific to a particular scenario component are described in the scenario component chapters.

14.1 Environment setup

This section describes the overall environment used for the scenario. It has been implemented on real machines and tested exactly as presented. The complete environment and scenario can be successfully set up and run using the information and materials supplied with this book.

The models presented for the logical topology and physical topology can be adapted to suit available hardware, software, and the scenario components that will be run, but it may require additional knowledge that is not covered by this book.

The following topics are discussed:

- The logical topology of the scenario environment
- The physical topology of the scenario environment
- Machines configuration and software installation

14.1.1 The logical topology of the scenario environment

Figure 14-1 shows the logical WebSphere MQ topology of the scenario environment.

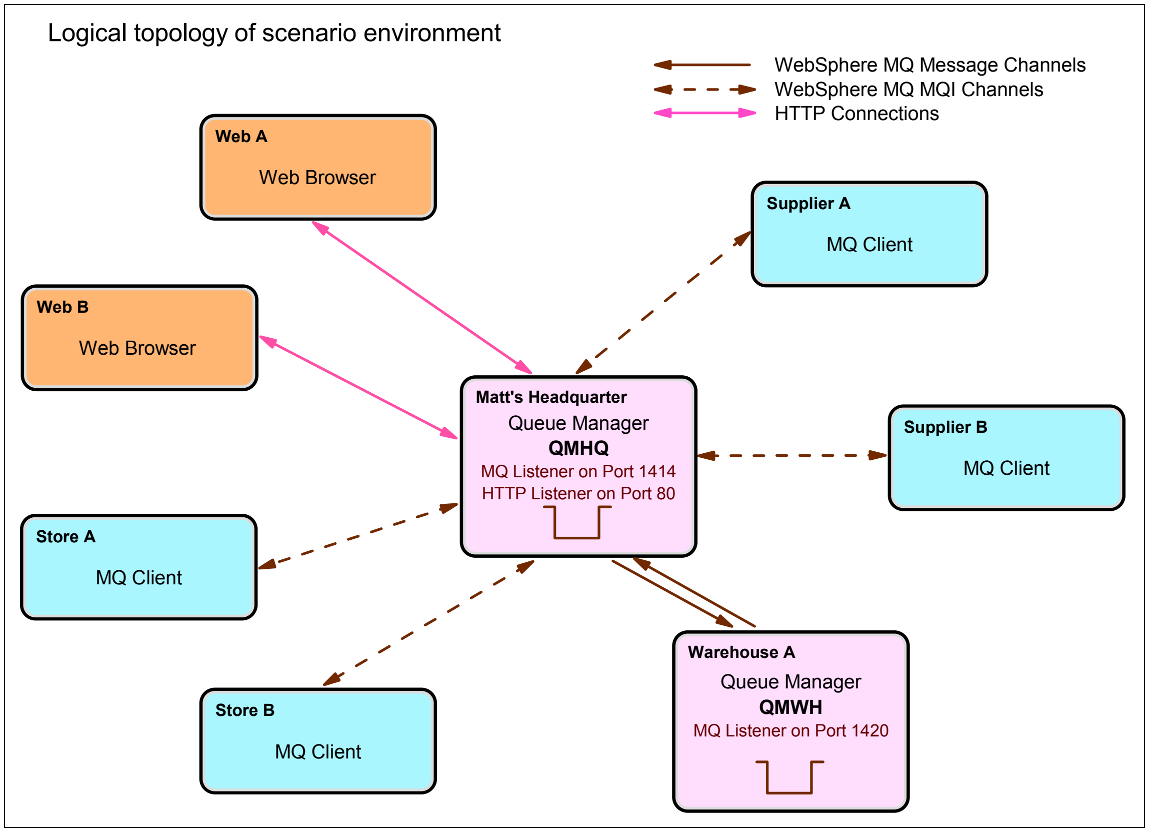


Figure 14-1 Logical topology of scenario environment

The environment consists of two MQ queue managers, four MQ Clients, and two systems that have a Web browser. All required queue managers and their connections are listed in Table 14-1.

Table 14-1 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

14.1.2 The physical topology of the scenario environment

Figure 14-2 shows the physical topology of our scenario's environment.

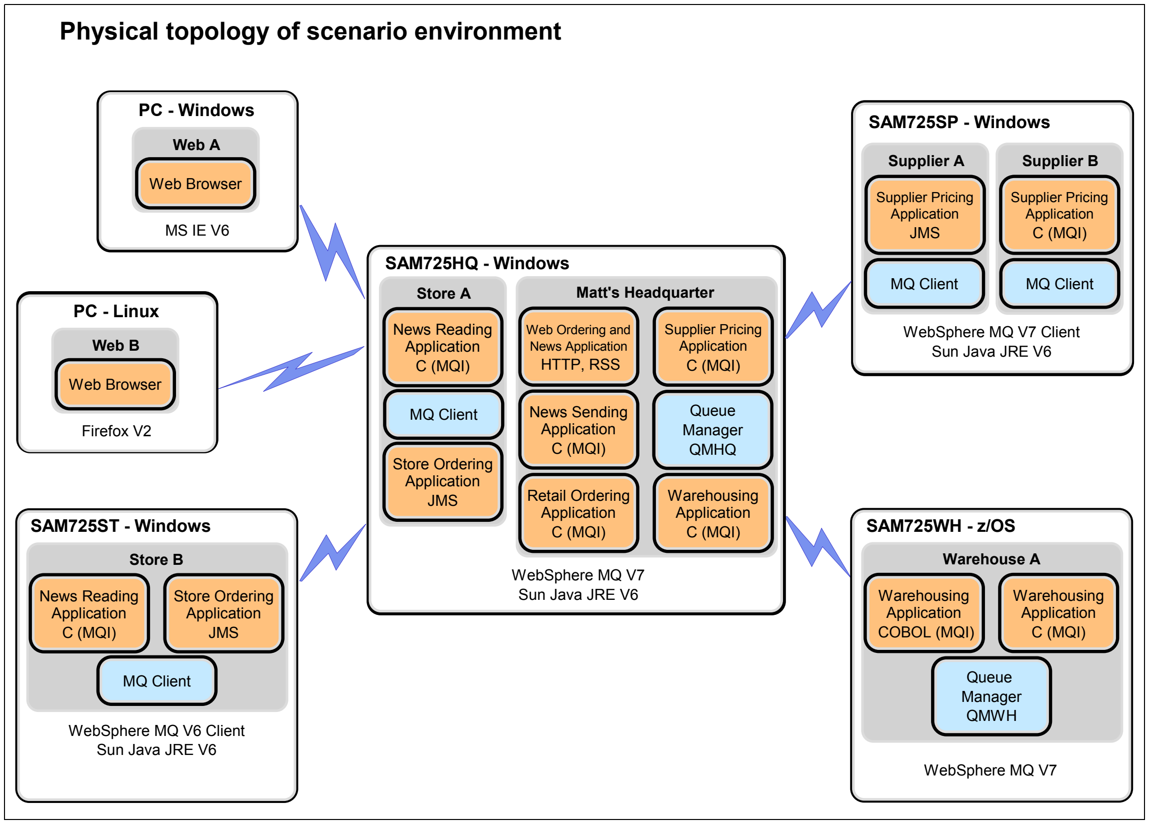

Physical topology of scenario environment
PC - Windows
Web A
Web Browser
MS IE V6
PC - Linux
Web B
Web Browser
Firefox V2
SAM725ST - Windows
Store B
News Reading Application C (MQI)
Store Ordering Application JMS
MQ Client
WebSphere MQ V6 Client
Sun Java JRE V6
SAM725HQ - Windows
Store A
News Reading 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 14-2 Physical topology of scenario environment

The physical environment consists of four machines that run Windows and two client PCs that run Windows and Linux operating systems. All software products used, and their versions, are listed in Table 14-2 through Table 14-7 on page 274.

Table 14-2 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
Application Server for running WebSphere MQ HTTP Bridge	WebSphere Application Server V6.1.0.0

Table 14-3 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 14-4 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 14-5 SAM725WH: z/OS machine for warehouse

Software	**Installed level**
Operating system	z/OS V1.9
WebSphere MQ	V7.0

Table 14-6 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 14-7 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 the machines listed above are not required to run the entire scenario. For example, it is possible to use only one machine and install all required components on it.

Scenario programs may be need to be recompiled to run on other platforms.

14.1.3 Machine configuration and software installation

This section describes machine configuration and installation of required software.

The installation of operating systems is not covered in this chapter. Appropriate machines with supported operating systems are a prerequisite for this scenario.

Note: For information about supported platforms for WebSphere MQ V7.0, refer to the *WebSphere MQ System Requirements*, at the IBM Web page:

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

Basic machine setup

The following tasks must be done as prerequisites for the next steps:

- Check the user permissions: Administrator permissions are needed for machine configuration and software installations.
- Set up TCP/IP address and host name: TCP/IP access is required among scenario machines for the MQ listener ports, as described in Table 14-1 on page 272 and Figure 14-2 on page 272.

Creating groups and user IDs

The groups required for our scenario must be created on the machines as described in Table 14-8.

Table 14-8 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 user's (Table 14-9) must be created on the machines and put into appropriate groups.

Table 14-9 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 users are a member of the mqm group, to satisfy proper security policy.

Tip: The scenario will operate without any demonstration of security by running all of the components from the mqm user or a user that is in the mqm group.

WebSphere MQ V7.0 product installation

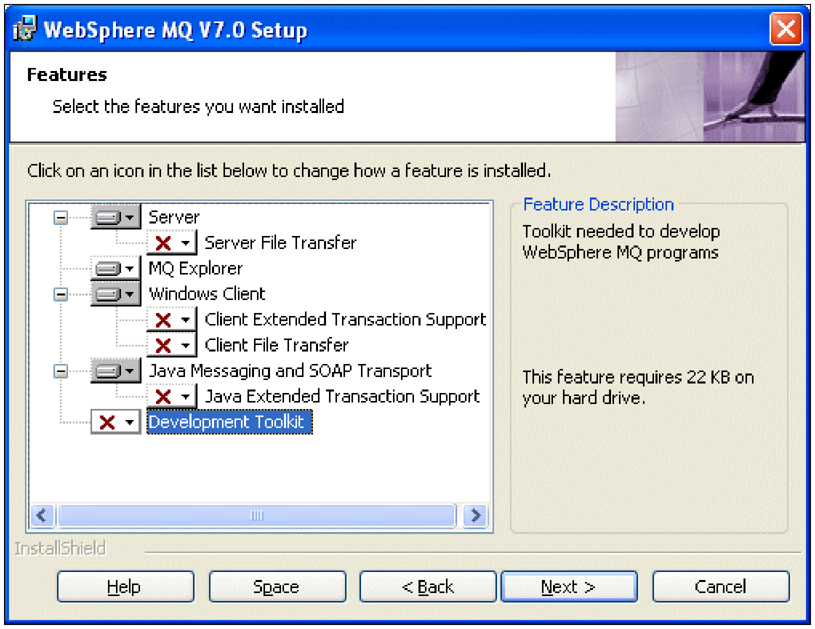
Perform the WebSphere MQ V7.0 product installation in accordance with the product installation guides, available in the WebSphere MQ V7.0 Information Center at:

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 WebSphere MQ server installation on the headquarters machine, SAM725HQ. Select the **Custom** installation and choose Server, MQ Explorer, Windows Client, and Java Messaging and SOAP Transport components, as depicted in Figure 14-3.



Features

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

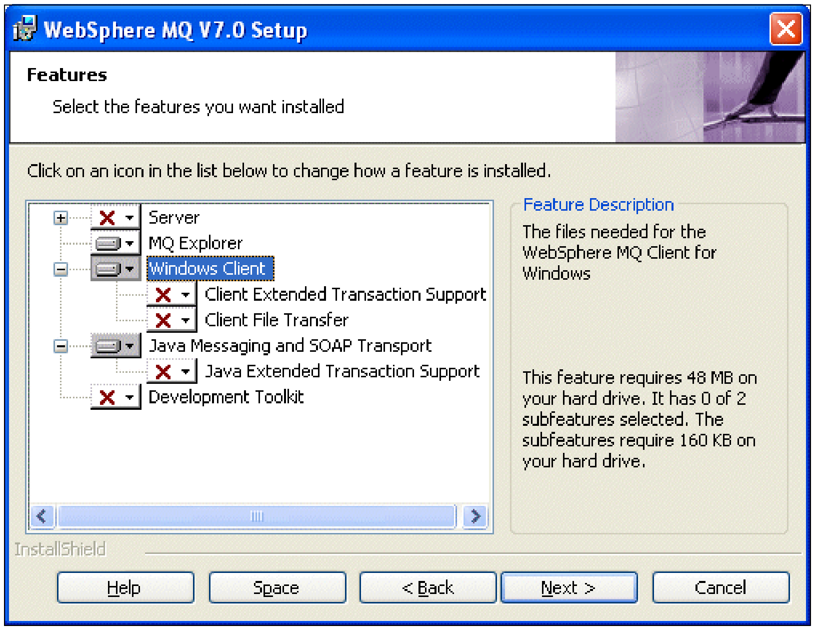
This feature requires 22 KB on your hard drive.

Figure 14-3 WebSphere MQ server installation

Use defaults for any other parameters or settings during the installation dialog.

WebSphere MQ client installation on suppliers machine

Perform WebSphere MQ client installation on supplier machine SAM725SP. Select the **Custom** installation and choose MQ Explorer, Windows Client, and Java Messaging and SOAP Transport components, as depicted in Figure 14-4.


WebSphere MQ V7.0 Setup

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 14-4 WebSphere MQ client installation

Use defaults for any other parameters or settings during the installation dialog.

WebSphere MQ server installation on warehouse machine

Perform WebSphere MQ server installation on the z/OS warehouse machine, SAM725WH. The server component installation is performed via the standard SMP/E installer.

WebSphere MQ V6.0 client installation on store B machine

Download and install WebSphere MQ SupportPac MQC6 WebSphere MQ V6.0 Clients on the store B machine, SAM725ST, which is available at:

`http://www.ibm.com/support/docview.wss?uid=swg24009961`

Select **Typical** installation and use the defaults for any other parameters or settings during the installation dialogs.

Java runtime environment installation

Download and install the latest Java runtime code (Version 1.6.0_03 at the time of writing), available at the 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.

WebSphere Application Server installation

To run the WebSphere MQ HTTP Bridge component of the scenario, install WebSphere Application Server V6.1 on the headquarters machine. It is possible to use the WebSphere MQ HTTP Bridge in an existing installation and setup of WebSphere Application Server or WebSphere Application Server Community Edition. Details of installation and setup of WebSphere Application Server and WebSphere Application Serve Community Edition are beyond the scope of this book.

Refer to Chapter 18, “Scenario: Web ordering over HTTP” on page 343, for instructions for deploying the bridge in WebSphere Application Server.

14.2 WebSphere MQ objects setup

This section contains lists of MQ objects required for our scenario:

- Queue managers
- Queue manager objects
  - Listeners
  - Channels
  - Queues
  - Topics and subscriptions
- Object Authority Manager (OAM)

Tip: All the queue managers listed below are not required to run the entire scenario. It is possible to use only one queue manager and create all required objects on it.

The scripts that are supplied for our scenario must be modified to create the required objects on only one queue manager.

14.2.1 Creating the queue managers

Two queue managers must be created using the appropriate operating system control commands, as listed in Table 14-10.

Table 14-10 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.

14.2.2 Creating the queue managers objects

The queue manager objects at headquarters and the warehouse must be created by running two MQSC command scripts. These scripts use the REPLACE parameter so they can be run repeatedly. Refer to Appendix A, “Scenario preparation scripts” on page 373, for printed copies.

To set up the scripts on the headquarters system:

1. Appendix B, “Additional material” on page 379, contains instructions to download the compressed file Chapter14_ScenarioPrep.zip.

   Tip: The compressed files for other scenario chapters that are going to be used should be downloaded at this stage. If all the chapters will used the compressed file, All.zip should be downloaded and extracted, as this file contains the sub-folders for all scenario chapters.

2. Create the folder C:\Scenario on the headquarters system. This is used to contain data and programs for all of the scenario chapters.
3. Extract Chapter14_ScenarioPrep.zip into the C:\Scenario folder.

This creates a sub-folder called Chapter14_ScenarioPrep containing the following two MQSC command script files:

- QMHQobjects.txt: Object definition MQSC commands for the headquarters queue manager. Issue the following command in a Windows command prompt window to run this script:

  ```
  runmqsc QMHQ < QMHQobjects.txt
  ```

  Tip: It is possible to verify commands in the script without performing them by specifying the -v parameter:

  ```
  runmqsc -v QMHQ < QMHQobjects.txt
  ```

- QMWHobjects.txt: Object definition MQSC commands for the warehouse queue manager. FTP this file to the z/OS mainframe system used by the warehouse and use the batch utility CSQUTIL to run the commands. If a different type of platform is being used for the warehouse, the command script can be executed using the **`runmqsc QMWH < QMWHobjects.txt`** command.

Tip: WebSphere MQ Explorer can be used instead of MQSC commands. Create all objects from the tables below using the appropriate MQ Explorer tasks.

The listeners

Listeners required for the entire scenario are listed in Table 14-11.

Table 14-11 List of listeners

QM name	Listener name	Port number	Controlled by
QMHQ	QMHQ.TCP	1414	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 all components of the scenario except the news scenario described in Chapter 17, “Scenario: News using client” on page 325.
- The MQ listener QMWH.TCP is related to the warehousing scenario described in Chapter 19, “Scenario: Warehousing using call back” on page 361.

Tip: Only the one MQ listener with port number 1414 is used if only one queue manager is used for the entire scenario.

The channels

Channels required for the entire scenario are listed in Table 14-12.

Table 14-12 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 19, “Scenario: Warehousing using call back” on page 361.
- The server-connection channels QMHQ_SUPP_A and QMHQ_SUPP_B are related to the supplier pricing scenario described in Chapter 15, “Scenario: Supplier pricing using Pub/Sub” on page 287.
- The server-connection channel QMHQ_STORES is related to the store ordering scenario described in Chapter 16, “Scenario: Store ordering with JMS” on page 303, and to the news scenario described in Chapter 17, “Scenario: News using client” on page 325.
- The server-connection channel QMHQ_NEWS is related to the news scenario described in Chapter 17, “Scenario: News using client” on page 325.

Tip: The sender and receiver channels are not needed if only one queue manager is used for the entire scenario.

The queues

Queues required for the entire scenario are listed in Table 14-13.

Table 14-13 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 19, “Scenario: Warehousing using call back” on page 361.
- The local queues MATT.RETAIL.ORDERS and MATT.RETAIL.RESPONSES are related to the store ordering scenario described in Chapter 16, “Scenario: Store ordering with JMS” on page 303, and to the Web ordering scenario described in Chapter 18, “Scenario: Web ordering over HTTP” on page 343.
- The local queue MATT.RETAIL.WH.ORDERS is related to:
  - The store ordering scenario described in Chapter 16, “Scenario: Store ordering with JMS” on page 303
  - The Web ordering scenario described in Chapter 18, “Scenario: Web ordering over HTTP” on page 343
  - The warehousing scenario described in Chapter 19, “Scenario: Warehousing using call back” on page 361
- The local queue MATT.RETAIL.WH.ORDERS is related to the supplier pricing scenario described in Chapter 15, “Scenario: Supplier pricing using Pub/Sub” on page 287.
- The remote and local queues MATT.WH.ORDERS are related to the warehousing scenario described in Chapter 19, “Scenario: Warehousing using call back” on page 361.

Tip: The transmission queues and the MATT.WH.ORDERS remote queue are not needed if only one queue manager is used for the entire scenario.

The topics and subscriptions

Topics and subscriptions required for the entire scenario are listed in Table 14-14 and Table 14-15.

Table 14-14 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 14-15 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 15, “Scenario: Supplier pricing using Pub/Sub” on page 287.
- The topic MATT.TOPIC.WAREHOUSE is related to the warehousing scenario described in Chapter 19, “Scenario: Warehousing using call back” on page 361.

- The topic MATT.TOPIC.RETAIL is related to:
  - The supplier pricing scenario described in Chapter 15, “Scenario: Supplier pricing using Pub/Sub” on page 287
  - The store ordering scenario described in Chapter 16, “Scenario: Store ordering with JMS” on page 303
  - The Web ordering scenario described in Chapter 18, “Scenario: Web ordering over HTTP” on page 343
- The topics MATT.TOPIC.NEWS.INTERNAL and MATT.TOPIC.NEWS.PUBLIC are related to the news scenario described in Chapter 17, “Scenario: News using client” on page 325.
- The subscription MATT.SUB.WH.ORDERS is related to the supplier pricing scenario described in Chapter 15, “Scenario: Supplier pricing using Pub/Sub” on page 287.

Note: There are no topic objects or subscriptions on the QMWH queue manager so there is no additional action required for using only one queue manager for the entire scenario.

14.2.3 Setting object authority

The appropriate MQ authorities for headquarters queue manager objects must be setup in OAM using the control command **`setmqaut`**. The Windows batch command file QMHQsetaut.bat contains the **`setmqaut`** commands to set the authorities, as listed in Table 14-16. The batch command file removes all authorities at first and can be run repeatedly.

Table 14-16 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

QMHQsetaut.bat is supplied in Chapter14_ScenarioPrep.zip, which was setup in 14.2.2, “Creating the queue managers objects” on page 279.

To run the command file:

1. On the headquarters Windows system, open a command prompt window.
2. Issue the command:

```
C:\Scenario\Chapter14_ScenarioPrep\QMHQsetaut
```

3. Check the displayed output to ensure that all the **`setmqaut`** commands executed successfully.

The appropriate authorities for warehouse queue manager objects must be set in RACF on z/OS. The RACF commands are not provided with this book.

Tip: The authority settings for warehouse queue manager QMWH are not needed if only one queue manager is used for the entire scenario.

15

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 in WebSphere MQ V7.0.

Durable and non-durable subscriptions are used with managed and non-managed destinations.

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.

This chapter contains the following sections:

15.1 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 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. When Matt's Deli needs to purchase stocks of products, this system broadcasts requests to all suppliers for price quotes. 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 and records the cheapest quote. Headquarters would then send a purchase order to the selected supplier and receive stock at the warehouse, but this 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 by publishing a *price quote* message to the topic "matt/supplierquote/*CatalogId*". The catalog ID is a topic string that is concatenated to "matt/supplierquote/".

Matt's Deli has an application program that subscribes to the topic "matt/supplierquote/cat/#" to receive quotes from all suppliers. This application receives the supplier quote messages. It checks whether the price is cheaper than a previous quote received from another supplier. It calculates a retail price (a 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/*CatalogId*". This topic represents the retail price catalog. If the previous quote is from the same supplier for the same product the new quote is accepted as a 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.

The retail price catalog is used by the stores to display products and prices before making purchase orders.

Figure 15-1 describes the programs and the interaction between headquarters and the suppliers.

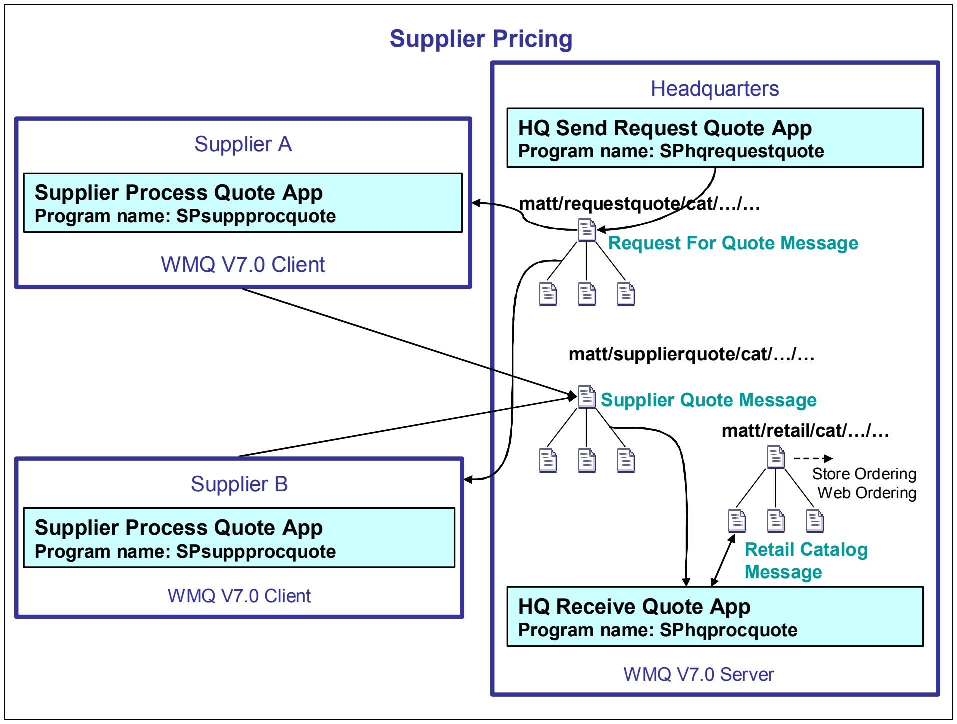

Supplier Pricing
Headquarters
HQ Send Request Quote App
Program name: SPhqrequestquote
Supplier A
Supplier Process Quote App
Program name: SPsuppprocquote
WMQ V7.0 Client
matt/requestquote/cat/…/…
Request For Quote Message
matt/supplierquote/cat/…/…
Supplier Quote Message
matt/retail/cat/…/…
Store Ordering
Web Ordering
Supplier B
Supplier Process Quote App
Program name: SPsuppprocquote
WMQ V7.0 Client
Retail Catalog Message
HQ Receive Quote App
Program name: SPhqprocquote
WMQ V7.0 Server

Figure 15-1 Supplier pricing component

15.2 Deploying the supplier pricing component

The supplier pricing component is made up of three application programs. Two of the programs run locally on the headquarters computer system and the third 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 programs implement the supplier pricing component:

- HQ send request for quote (SPhqrequestquote): This 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 system to receive supplier quotes, to decide whether 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 Publish/Subscribe services for the supplier pricing component. The headquarters application programs run on the same server as the queue manager and they connect to the queue manager using server bindings. The supplier programs are MQ Client applications and they connect to the queue manager using client bindings over a communications network.

15.2.1 HQ send request for quote program

This 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 the command-line argument 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 standard input (the keyboard). 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 calls 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 by concatenating the topic string specified in the topic object with the topic string received from the keyboard. It uses the MQPUT1 MQI call to publish to a topic.

15.2.2 HQ process supplier quotes program

This C language program runs on the same server as the queue manager. The main functions of this program are:

- It receives two optional command arguments. The first is the subscription queue name and the second 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 that this program is executed and it remains active until it is removed by the administrator using MQ Explorer or a MQSC 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 the function ProcessAQuote is called to do the following processing:

  a. 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 the receiving of retained publications.

  b. 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 publication is expected.

  c. If a retained publication is not received then this is a new product. Therefore, a PublishRetailCatalogItem function 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/*ProductCategory*/*ProductType*/*ProductName*". 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.

   d. 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.
   e. 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 by calling the function PublishRetailCatalogItem.
   f. 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.
   g. If the current price is lower than or equal to the price quoted then the supplier quote is discarded and the retail price catalog is not updated.

- For every retail price catalog update the function PublishRetailCatalogItem is called:
   a. It opens a topic object with a topic string that represents the catalog ID of the product.
   b. It prepares a retail price publication. This message is comma-separated text with five fields:
      - Catalog ID
      - Product description
      - Retail price
      - Supplier price
      - Supplier name
   c. It publishes the retail price message as a retained publication by 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.

15.2.3 Supplier process quotes program

This is a C language program that runs as a MQ Client application in the supplier's computer system. This program interacts with the supplier's staff to receive and respond to the Matt's Deli request for quotes. The main functions of this program are:

- It receives a queue manager name as the command argument 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=*SvrconnChannelName*/TCP/*Host*(*Port*) must be defined before running this program.
- It creates a subscription using a topic object called MATT.TOPIC.REQUESTQUOTE and an object string "cat/#". The resolved topic string is the concatenation of the topic string defined in the topic object and the 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 whether it should continue or terminate the program.
- It calls the function ProcessARequestQuote for each message received. This function does the following processing:
  a. It parses the received message to get the catalog ID and the production description.
  b. It displays to the user the request for the quote received.
  c. It asks the user for a price to quote, or to enter 0 (zero) if the request is to be ignored.
  d. If zeros is entered the function returns without doing any further process to the request for quote.
  e. If a price is entered then a supplier quote message is prepared.
  f. It opens a topic object called MATT.TOPIC.SUPPLIERQUOTE and an object string that is the catalog ID received in the request for quote.
  g. It publishes a supplier quote to the topic using MQPUT1.
- It loops to get next request for quote message from suppliers.

15.2.4 Required MQ objects

The supplier pricing component requires the following objects to be defined by the queue manager administrator:

- MATT.TOPIC.REQUESTQUOTE: This topic object relates to the topic “matt/requestquote”. The purpose of this object is to define access control such that Matt’s Deli headquarters can publish and subscribe to the topic but suppliers can only subscribe.
- MATT.TOPIC.SUPPLIERQUOTE: This topic object relates to the topic “matt/supplierquote” and defines that publications to this topic should be persistent by default. The access control allows suppliers to publish but not to subscribe to this topic. Headquarters should be able to subscribe and publish.
- MATT.SUPPLIER.QUOTES: This is the subscriber local queue for the non-managed durable subscription created by the HQ process supplier quotes program. Supplier quote publications are queued while the program is busy or not active.

Example 15-1 contains the MQSC commands to define these objects.

Example 15-1 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)
```

15.2.5 Installation of supplier pricing component

The supplier pricing component programs are packaged in a file called Chapter15_SupplierPricing.zip available for download in Appendix B, “Additional material” on page 379.

To install the executable code:

1. Extract the compressed file into the C:\Scenario directory on the headquarters machine.
2. The following directories are created to contain each program:
   - C:\Scenario\Chapter15_SupplierPricing\SPhqprocquote
   - C:\Scenario\Chapter15_SupplierPricing\SPhqrequestquote
   - C:\Scenario\Chapter15_SupplierPricing\SPsuppprocquote

   There are two additional directories with other programs that can be used to test this component:

   - C:\Scenario\Chapter15_SupplierPricing\loader
   - C:\Scenario\Chapter15_SupplierPricing\SPsuppsendquote

   Each directory contains the following files:

   - SPrun*ProgramName*.bat: This is a Windows command file that executes the program.
   - *ProgramName*.c: This is the program source code.
   - *ProgramName*.exe: This is the executable code linked with the MQ server bindings library.
   - *ProgramName*c.exe: This is the executable code linked with the MQ client bindings library.
   - make.bat: This batch command file 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.

15.3 Running the supplier pricing component

To run the supplier pricing component programs, execute each program in its own command prompt window.

15.3.1 HQ send request for quote program

To run the SPhqrequestquote program:

1. Open a Windows command prompt and execute the following command:

   ```
   C:\> runas /user:hquser cmd.exe
   ```

2. Select the command prompt window started by the **runas** command.
3. Check that the queue manager is running:

   ```
   C:\> dspmq
   QMNAME(QMHQ) STATUS(Running)
   ```

4. Change the directory to the location of the executable code:

   ```
   C:\> cd C:\Scenario\Chapter15_SupplierPricing\SPhqrequestquote
   ```

5. Execute the program (or execute the command file SPrunhqrequestquote.bat):

   ```
   C:\Scenario\Chapter15_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 :
   ```

6. Enter the catalog ID for the product to request the quote:

   ```
   --> CatalogId : cat/fresh/fruit/oranges

   Enter product description:
   ..  (note no commas)

   --> Description :
   ```

7. 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 :
```

8. 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 *******************************
```

15.3.2 HQ process supplier quotes program

To run the SPhqprocquote program:

1. Open a Windows command prompt and execute the following command:

```
C:\> runas /user:hquser cmd.exe
```

2. Select the command prompt window started by the **runas** command.
3. Check that the queue manager is running:

```
C:\> dspmq
QMNAME(QMHQ) STATUS(Running)
```

4. Change the directory to the location of the executable code:

```
C:\> cd C:\Scenario\Chapter15_SupplierPricing\SPhqprocquote
```

5. Execute the program (or execute the command file SPrunhqprocquote.bat):

```
C:\Scenario\Chapter15_SupplierPricing\SPhqprocquote>
SPhqprocquote.exe
Supplier Pricing Process Supplier Quote Application START
<*> Connected to Queue Manager: QMHQ
<*> Open Subscription queue: <MATT.SUPPLIER.QUOTES>
<*> MQSUB ResolvedTopicString <matt/supplierquote/cat/#>
<*> Calling MQGET : 180 seconds wait time
```

6. 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
```

7. 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
```

15.3.3 Supplier process quotes program

To run the SPsuppprocquote program:

1. Open a Windows command prompt and execute the following command:

   ```
   C:\> runas /user:suppa cmd.exe
   ```

2. Select the command prompt window started by the **runas** command.
3. Change the directory to the location of the executable code:

   ```
   C:\> cd C:\Scenario\Chapter15_SupplierPricing\SPsuppprocquote
   ```

4. Set the MQSERVER environment variable to establish the connection information to the queue manager:

   ```
   C:\Scenario\Chapter15_SupplierPricing\SPsuppprocquote> set MQSERVER=
   QMHQ_SUPP_A/tcp/9.20.16.251(1414)
   ```

5. Execute the program (or execute command file SPrunsuppprocquote.bat):

   ```
   C:\Scenario\Chapter15_SupplierPricing\SPsuppprocquote>
   SPsuppprocquotec.exe QMHQ
   **************************************************************
   * Welcome to MATTS DELI receive request for quote          ***
   * and send a supplier price quote.                         ***
   **************************************************************
   Enter your Supplier Identification
   .. or press ENTER to end

   --> SupplierId :
   ```

6. Enter the supplier identification:

   ```
   --> SupplierId : SUPPA
   <*> Connecting to queue manager: QMHQ
   <*> MQSUB ResolvedTopicString <matt/requestquote/cat/#>
   <*> Calling MQGET : 180 seconds wait time
   ```

7. 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 :
   ```

8. Enter a price to send the 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
```

9. 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 ********************************
```

15.4 Verifying the supplier pricing component

To verify the supplier pricing component:

1. Start programs SPhqrequestquote and SPhqprocquote in separate command windows in the headquarters server where the queue manager is running.
2. Start the program SPsuppprocquote on a command window in the supplier system as the MQ Client application.
3. Enter a request for quote on the SPhqrequestquote program.
4. Receive the request on the SPsuppprocquote program. Respond generating a supplier quote entering a price that is not zeros.
5. Observe the supplier quote being processed by the SPhqprocquote program.
6. 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
```

15.5 Summary

The Matt’s Deli supplier pricing component uses Publish/Subscribe to broadcast requests 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 the retained publications of WebSphere MQ V7.0 Publish/Subscribe to implement the retail price catalog. Retained publications allow a single price to be maintained 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 catalog is updated by publishing a new retained publication to the topic that represents 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.

16

Scenario: Store ordering with JMS

This component of our scenario illustrates the use of Java Message Service (JMS) to perform Publish/Subscribe and point-to-point messaging with an emphasis on the enhancements in WebSphere MQ V7.0.

Retained publications are used to store the retail catalog of all products offered by Matt's Deli. The stores use non-durable subscriptions with a wildcard in the topic string to browse publications on the topic tree.

This chapter includes the following sections:

16.1 Design overview

The product catalog required for the stores to place orders is maintained at the headquarters of Matt's Deli. The details of each product in the catalog is held as a retained publication on the WebSphere MQ queue manager at headquarters.

The store ordering application is independent of all other components of the scenario. However, the retail order processing application that runs at headquarters is also used by the Web ordering component, and it provides a feed to the warehousing component.

The product catalog must be pre-loaded with descriptions and pricing information by storing retained publications in the "matt/retail/cat" topic tree of the headquarters queue manager. Refer to 16.4, "Deploying the store application" on page 308, for details.

When a staff member at one of Matt's Deli stores starts the store ordering application at their local store, it first subscribes to the "matt/retail/cat/+" topic to fetch all the main product categories. The user then has the option to navigate down through the product hierarchy to select an item to order. The user can then submit an order for a product to the retail order application that is running at headquarters.

The headquarters application sends an order confirmation response back to the user for display. The store ordering application waits for a maximum of 3 seconds for the response and then control is returned to the user to place the next order. The user has an option to check for asynchronous responses from headquarters that were not received within the 3-second limit.

When the retail order application processes an order it also sends a point-to-point message to a queue, which is processed by the warehouse to fulfill delivery of the products ordered from the warehouse to the store.

The store ordering application and the retail order application both use Java Message Service for communicating via the MQ client to the headquarters queue manager.

Figure 16-1 shows the workflow of the store ordering and retail order applications.

Note: The headquarters application is also used in Chapter 18, “Scenario: Web ordering over HTTP” on page 343, and it generates messages that are used by the Warehousing component described in Chapter 19, “Scenario: Warehousing using call back” on page 361.

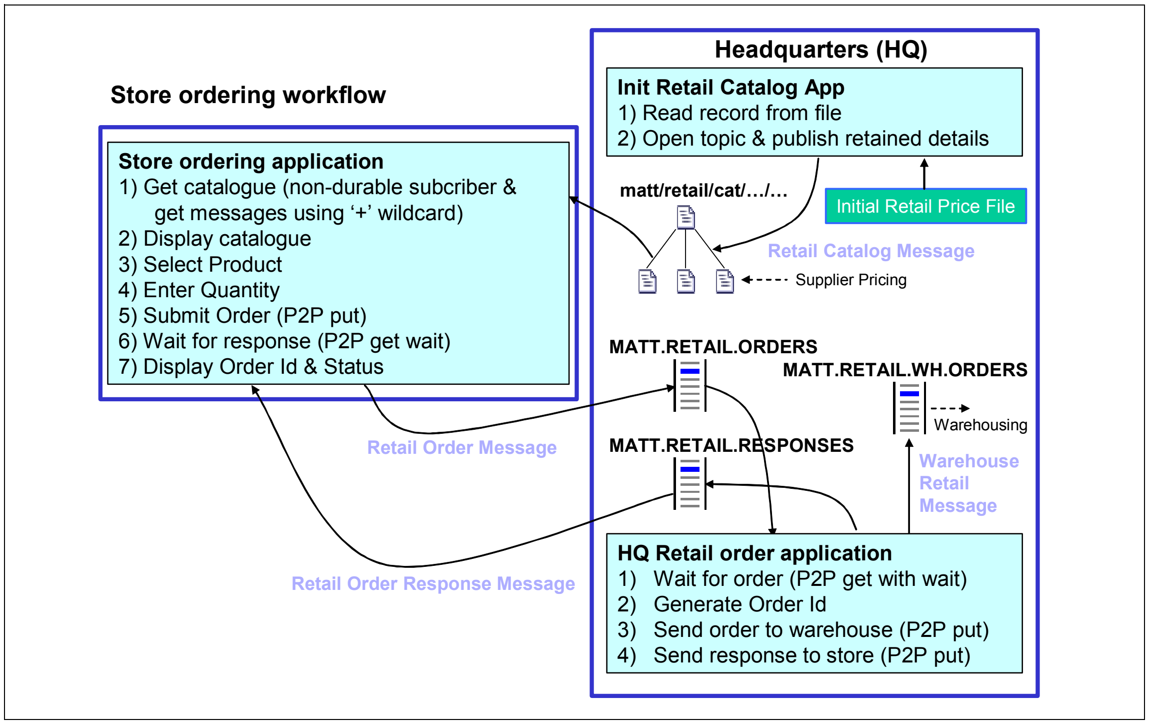

Store ordering workflow
Store ordering application
1) Get catalogue (non-durable subcriber & get messages using ‘+’ wildcard)
2) Display catalogue
3) Select Product
4) Enter Quantity
5) Submit Order (P2P put)
6) Wait for response (P2P get wait)
7) Display Order Id & Status
Headquarters (HQ)
Init Retail Catalog App
1) Read record from file
2) Open topic & publish retained details
matt/retail/cat/…/…
Initial Retail Price File
Retail Catalog Message
Supplier Pricing
MATT.RETAIL.ORDERS
MATT.RETAIL.WH.ORDERS
Warehousing
MATT.RETAIL.RESPONSES
Warehouse Retail Message
Retail Order Message
Retail Order Response Message
HQ Retail order application
1) Wait for order (P2P get with wait)
2) Generate Order Id
3) Send order to warehouse (P2P put)
4) Send response to store (P2P put)

Figure 16-1 Store ordering component

16.2 Deploying the headquarters application

The retail order application at headquarters uses JMS and must always be running and connecting to the headquarters queue manger. Before it can be started, the following prerequisites must be met:

- Ensure that the WebSphere MQ client is installed on the machine.
- Ensure that Java Development Kit 1.4.2 or later is installed and configured on the machine.

- The classpath variable must be set to include all the WMQ v7.0 jar files found under the *WMQv7_Home*/java/lib directory, as shown in Example 16-1.

Example 16-1 Set classpath variable

```
set classpath=.;C:\Program Files\IBM\WebSphere
MQ\Java\lib\CL3Export.zip;C:\Program Files\IBM\WebSphere
MQ\Java\lib\CL3Nonexport.zip;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqbind.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\connector.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\fscontext.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\jndi.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\jta.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\ldap.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\mqjbdf02.dll;C:\Program Files\IBM\WebSphere
MQ\Java\lib\mqjbnd05.dll;C:\Program Files\IBM\WebSphere
MQ\Java\lib\MQXAi02.dll;C:\Program Files\IBM\WebSphere
MQ\Java\lib\providerutil.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\rmm.jar;
```

- The Chapter16_StoreOrdering.zip file is supplied with the materials in Appendix B, “Additional material” on page 379. Extract the compressed file into the C:\Scenario directory. This creates the Chapter16_StoreOrdering sub-directory, which also contains several subdirectories.

Note: The HQProcess subdirectory contains all the files required for the headquarters application to run:

- hqprocess.jar: The main jar file that contains the class files
- HQSetup.bat: Initial setup file for the headquarters application
- HQRun.bat: Command file to execute store ordering application in headquarters

- Edit the HQSetup.bat file and modify the following values:
  - WMQv7_HOME: Set this value to the WebSphere MQ installed directory.

Example 16-2 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 16-3 Queue manager

  ```
  HQQUEMGRNAME=QMHQ
  ```

  – HQPORT: The port number on which the headquarter queue manager listens.

  Example 16-4 Port

  ```
  HQPORT=1414
  ```

  – HQHOSTNAME: The host name on which the headquarters queue manager resides.

  Example 16-5 Host name

  ```
  HQHOSTNAME=sam725hq
  OR
  HQHOSTNAME=12.87.234.65
  ```

- At the command prompt in the C:\Scenario\Chapter16_StoreOrdering\HQProcess directory, run HQSetup.bat:

  Example 16-6 Run HQSetup.bat

  ```
    HQSetup
  ```

Note: hqsetup.bat uses the WebSphere MQ JMSAdmin tool to create the required Java Naming and Directory Interface objects. Ensure that the JMSAdmin.config file under the *WMQv7_Home*/java/bin directory is configured correctly. For more information about configuring the JMSAdmin.config file, refer to the section “Using the WebSphere MQ JMS administration tool” in the *Using Java* manual, which is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

16.3 Running the headquarters application

To invoke the retail order application:

1. In a command prompt window, execute the following command:

   ```
   C:\runas /user:hquser cmd.exe
   ```

2. Select the command prompt window started by the **`runas`** command.
3. At the command prompt in the C:\Scenario\Chapter16_StoreOrdering\HQProcess directory, run HQRun.bat.

   Example 16-7 Execute HQRun.bat

   ```
   HQRun
   ```

   OR

   ```
   java -cp "./hqprocess.jar;%CLASSPATH%"
   com.ibm.residency.HQOrderProcessing
   -icf=com.sun.jndi.fscontext.RefFSContextFactory
   -url=file:/C:/JMSBindings
   ```

   Note: Enter the same value for -icf and -url as you would specify in the JMSAdmin.config file under the *WMQv7_Home*/java/bin directory.

4. Once the application starts it is ready to process the requests from the stores and Web ordering applications.

16.4 Deploying the store application

Before the store ordering application can be started, the following prerequisites must be met on each store machine:

1. Ensure that the WebSphere MQ client is installed on the machine.
2. Ensure that Java Development Kit 1.4.2 or later 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 *WMQv7_Home*/java/lib directory, as shown in Example 16-8.

Example 16-8 Include WebSphere MQ V7 jar files

```
set classpath=.;C:\Program Files\IBM\WebSphere
MQ\Java\lib\CL3Export.zip;C:\Program Files\IBM\WebSphere
MQ\Java\lib\CL3Nonexport.zip;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mq.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqbind.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\connector.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\fscontext.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\jms.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\jndi.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\jta.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\ldap.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\mqjbdf02.dll;C:\Program Files\IBM\WebSphere
MQ\Java\lib\mqjbnd05.dll;C:\Program Files\IBM\WebSphere
MQ\Java\lib\MQXAi02.dll;C:\Program Files\IBM\WebSphere
MQ\Java\lib\providerutil.jar;C:\Program Files\IBM\WebSphere
MQ\Java\lib\rmm.jar;
```

4. Extract the Chapter16_StoreOrdering.zip file to the C:\Scenario directory.

Note: The StoreOrdering subdirectory contains all the files required for the store ordering application to run:

- storeordering.jar: The main jar file that contains the class files
- SOSetup.bat: Initial setup file for the store ordering application
- SORun.bat: Command file to run the store ordering application on the store

5. Edit the SOSetup.bat file and modify the following values:

   – WMQv7_HOME: Set this value to the WebSphere MQ installed directory.

   Example 16-9 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 16-10 Queue manager

```
SOQUEMGRNAME=QMHQ
```

– SOPORT: The port number on which the head quarter queuemanager listens.

Example 16-11 Port

```
SOPORT=1414
```

– SOHOSTNAME: The host name on which the headquarters queue manager resides.

Example 16-12 Host name

```
SOHOSTNAME=sam725hq
```

OR

```
SOHOSTNAME=12.87.234.65
```

6. At the command prompt in the C:\Scenario\Chapter16_StoreOrdering\StoreOrdering directory, run SOSetup.bat:

```
SOQSetup
```

Note: SOSetup.bat file uses the WebSphere MQ JMSAdmin tool to create the required Java Naming and Directory Interface objects. Ensure that the JMSAdmin.config file under the *WMQv7_Home*/java/bin directory is configured correctly. For more information about configuring the JMSAdmin.config file, refer to the section “Using the WebSphere MQ JMS administration tool” in the *Using Java* manual, which is available in the WebSphere MQ V7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

16.5 Invoking the store application

To invoke the store ordering application:

1. In a command prompt window, execute the following command:

   ```
   C:\runas /user:stora cmd.exe
   ```

2. Select the command prompt window started by the **`runas`** command.
3. At the c:\Scenario\StoreOrdering prompt, execute SORun.bat, as in Example 16-13.

   Example 16-13 Execute SORun.bat

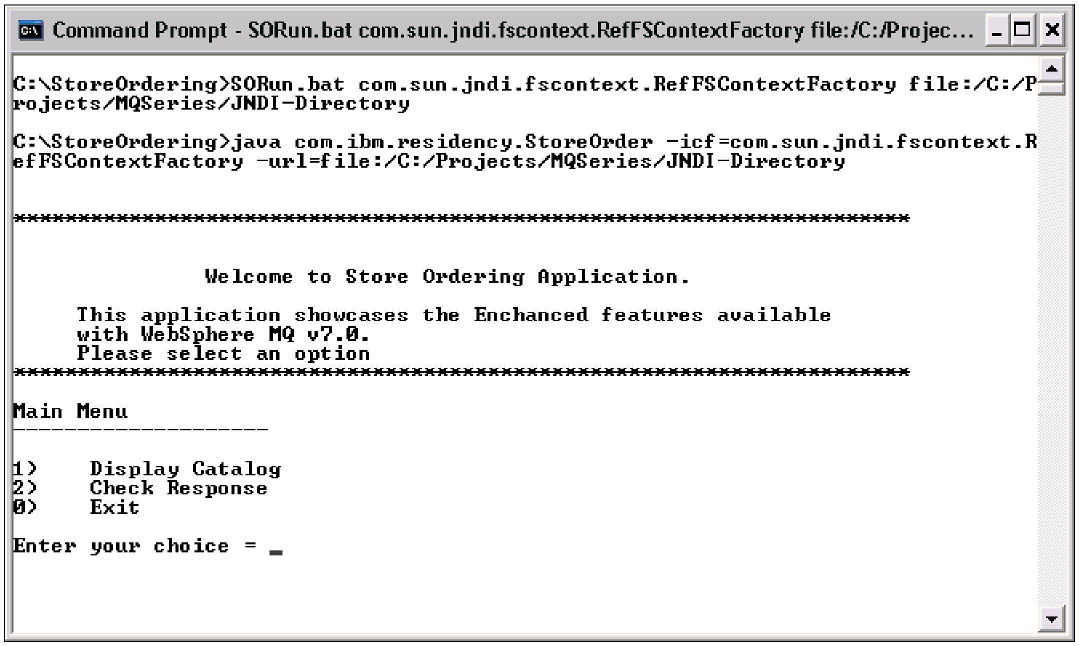
   ```
   SORun
   ```

   OR

   ```
   java -cp "./storeordering.jar;%classpath%"
   com.ibm.residency.StoreOrder
   -icf=com.sun.jndi.fscontext.RefFSContextFactory
   -url=file:/C:/JMSBindings
   ```

   Note: Enter the same value for -icf and -url as you would specify in the JMSAdmin.config file under the *WMQv7_Home*/java/bin directory.

4. Once the application starts the user is presented with the panel shown in Figure 16-2.


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 = _
```

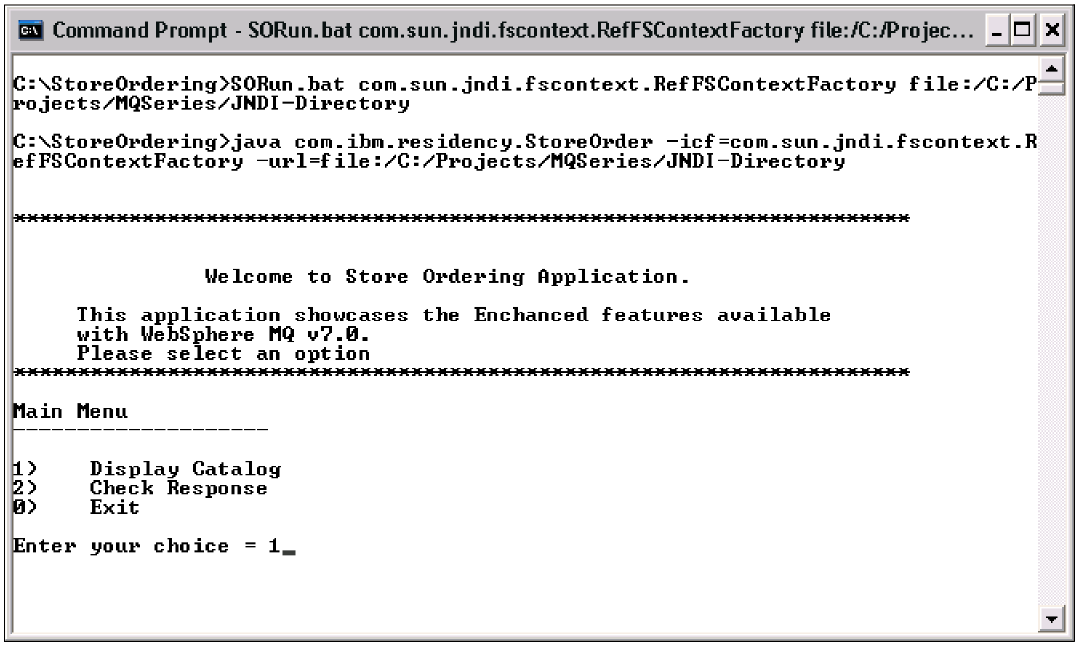
Figure 16-2   Store order application

16.6 Running the store application

To run the store application:

1. When the store ordering application is running at the main menu, the user is presented with the options shown in Figure 16-2:
   - Display Catalog: This option allows the user to view the product catalog and then submit orders.
   - Check Response: The user can use this option to view the responses for the previously submitted orders.
   - Exit: Terminate the application.

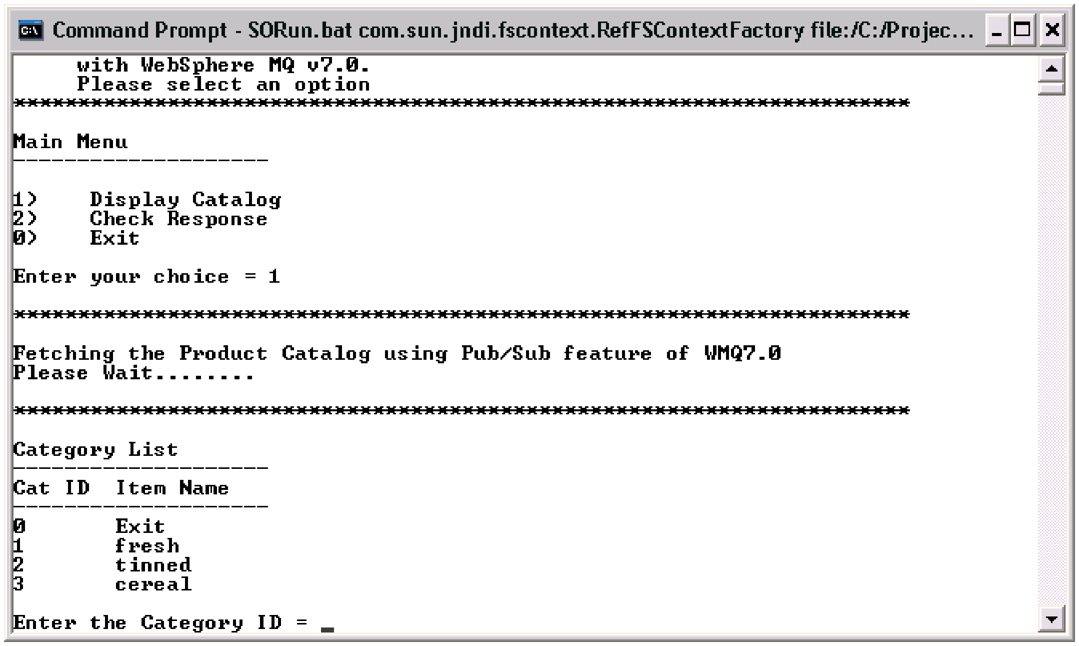
2. Display the product catalogue by entering 1 (Figure 16-3).


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 16-3   Display product catalog

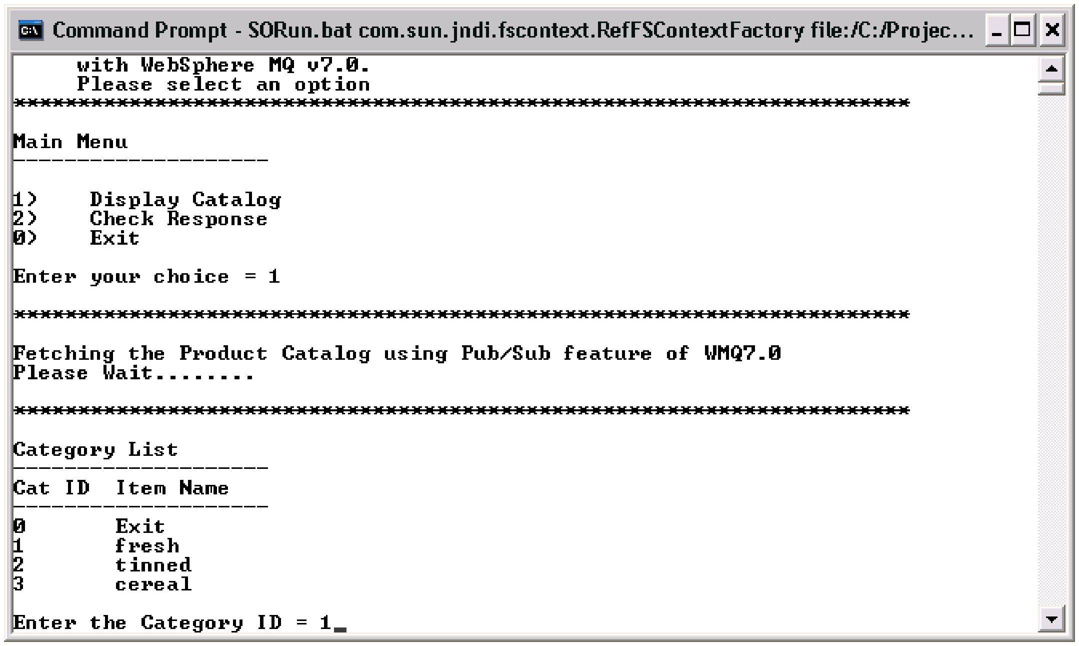
3. The user is then presented with a list of product categories (Figure 16-4).


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 16-4   Product categories

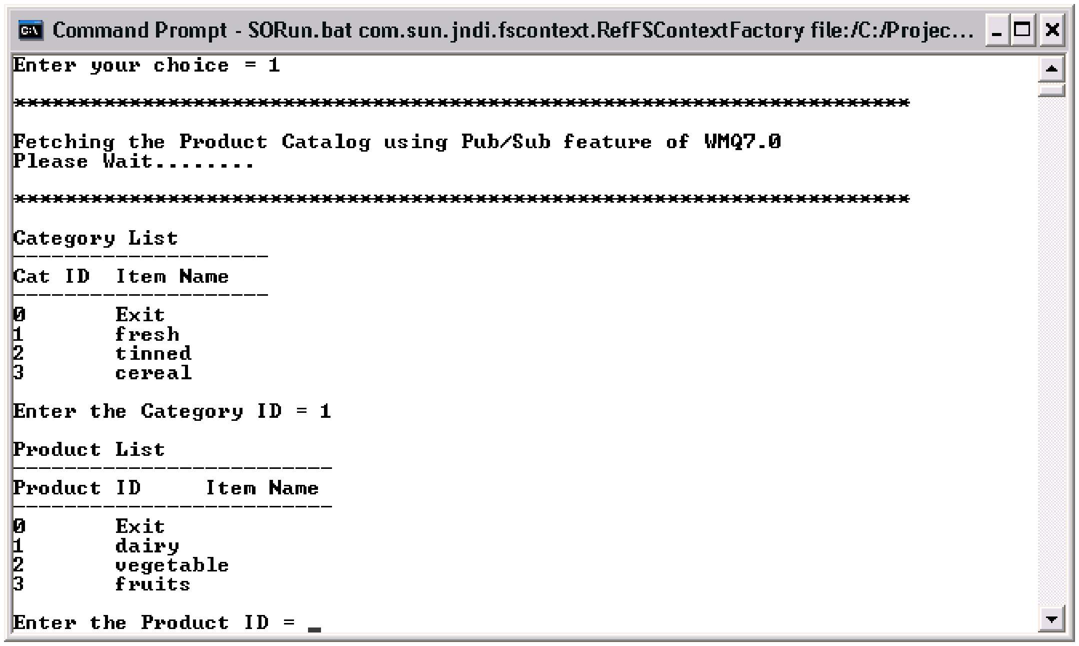
4. The user can select any of the product categories displayed to place an order. To select category fresh from the list, enter 1 (Figure 16-5).


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 16-5   Select categories

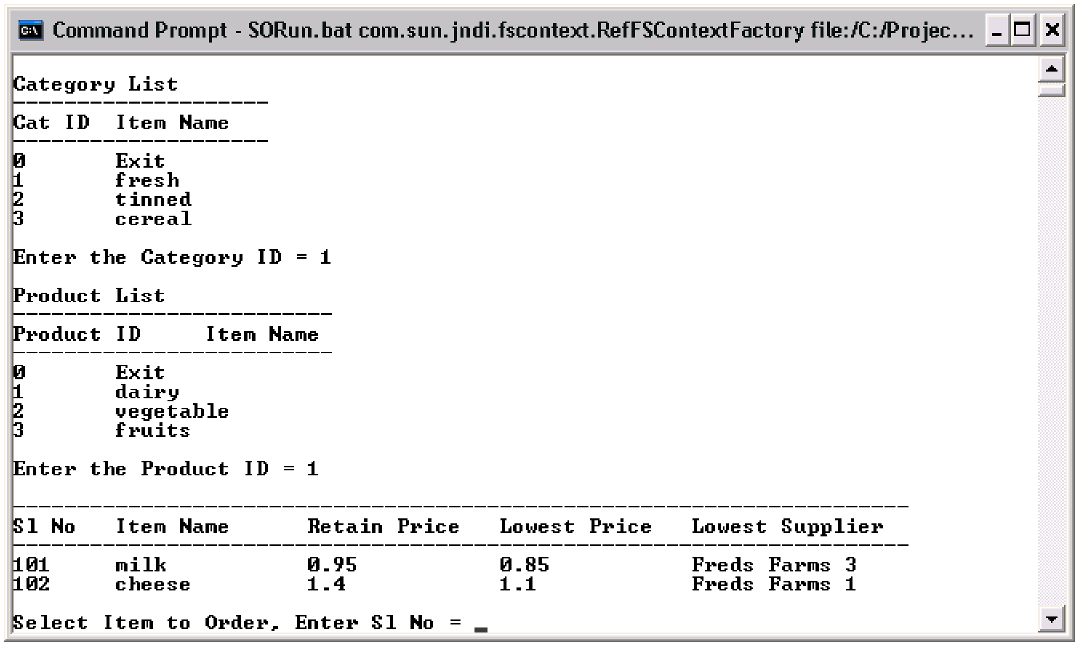
The user is presented with a list of products, as shown in Figure 16-6.


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 16-6 Product types

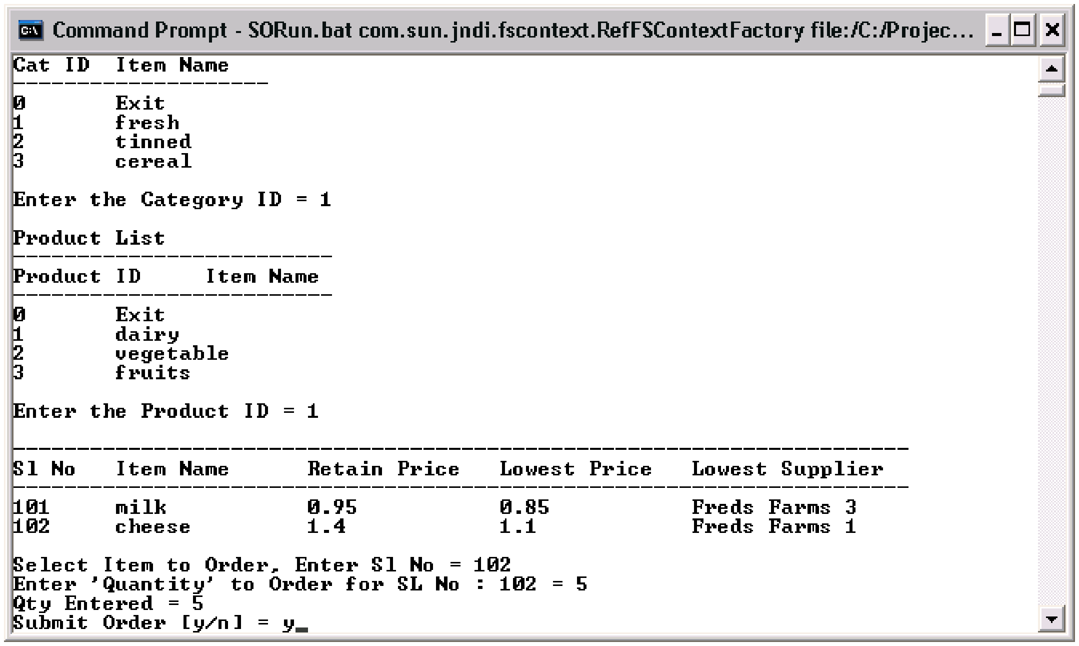
5. To view the different products under the product list, enter the corresponding number. To view products under the dairy type, enter `1`. The user is presented with the products shown in Figure 16-7.


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 16-7   Select product types

6. The user can now place the order for a product. Enter the Sl No, enter the quantity, and enter y to confirm submission of the order (Figure 16-8).

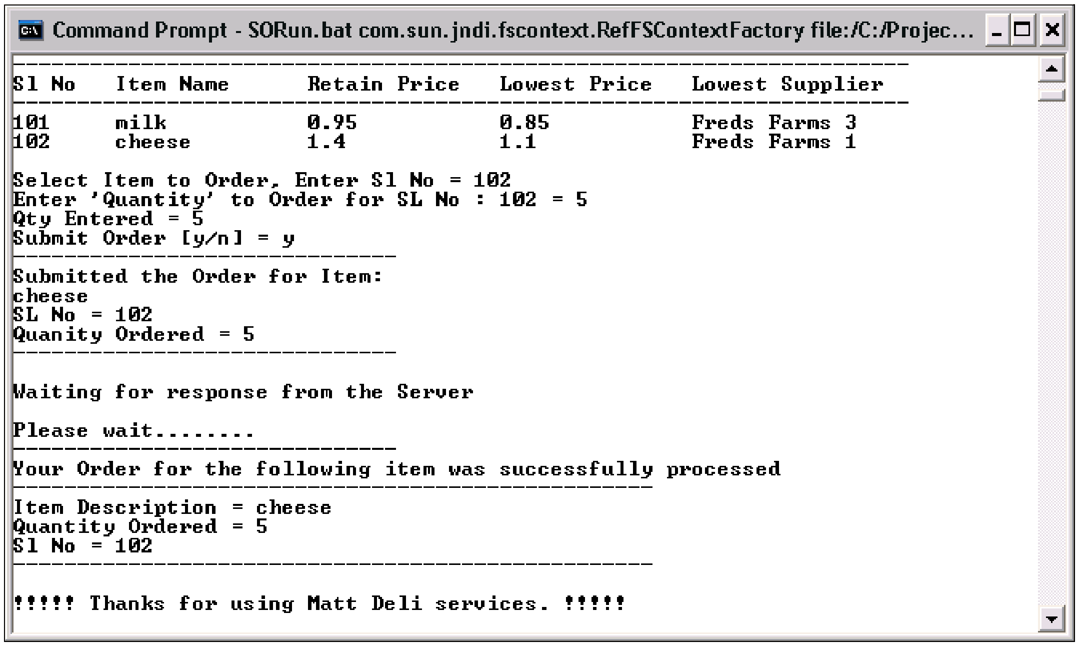


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
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 16-8   Submit order

7. The headquarters application sends a response back to the user indicating whether the order was successful, as shown in Figure 16-9.

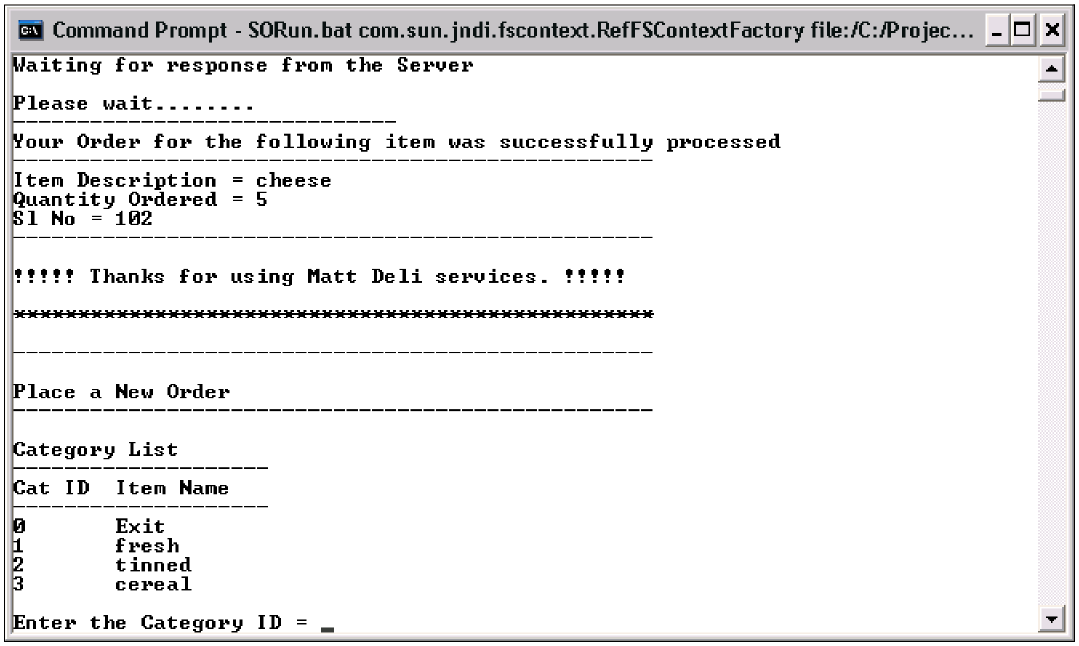

Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
----------------------------------------------------------------------
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 16-9   Order response

Note: Figure 16-9 shows that the order for the product cheese was submitted by the store to headquarters and a response was received. Headquarters also forwards this order to the warehouse.

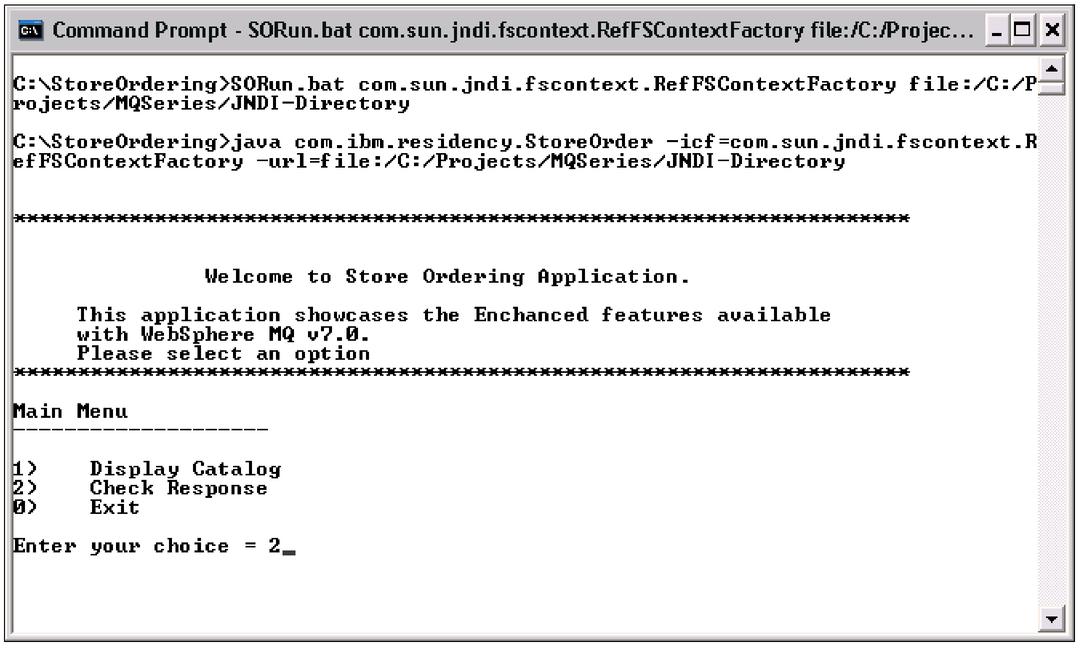
8. The store ordering application then allows the user to place a new order (Figure 16-10).


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 16-10 Place a new order

9. Check responses. In case headquarters was not able to send the response to the store within 3 seconds, the user still is able to check for the response asynchronously. On the main menu, select the Check Responses option by entering 2, as shown in Figure 16-11.

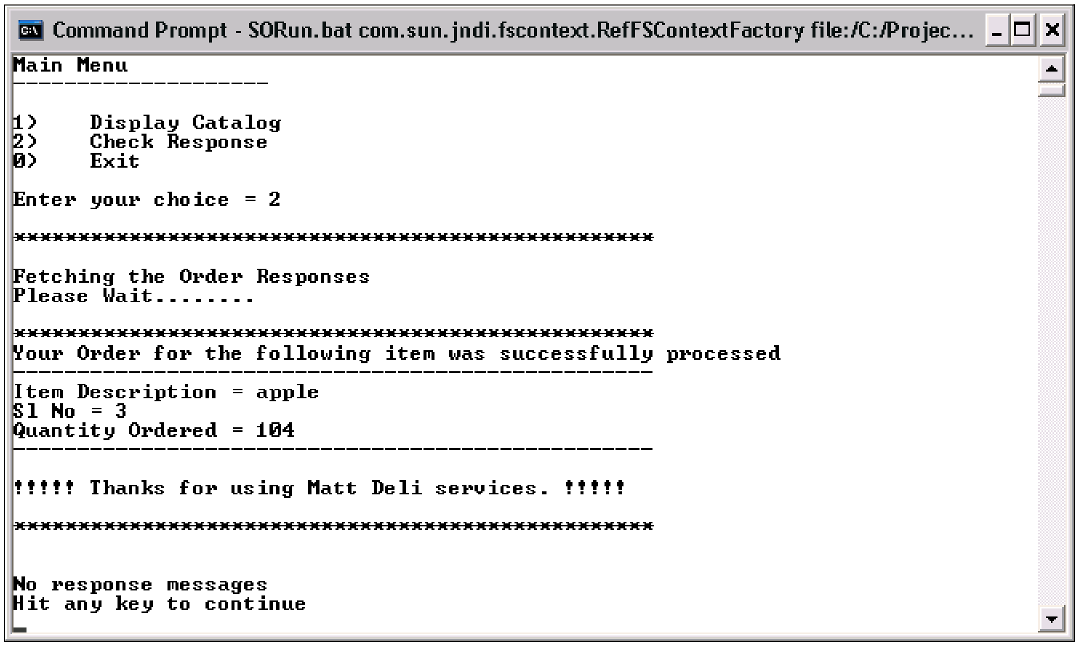


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
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 16-11   Select Check Response

10. The application then displays the responses for the previous orders submitted, as shown in Figure 16-12.

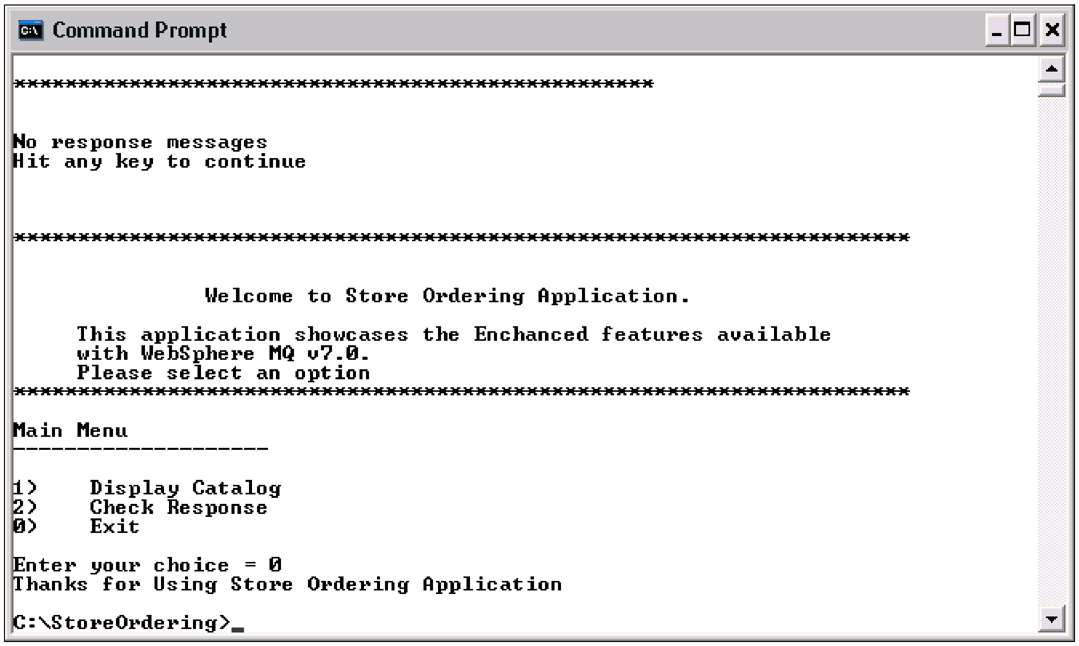


Command Prompt - SORun.bat com.sun.jndi.fscontext.RefFSContextFactory file:/C:/Projec...

```
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 16-12 Order status

11. The user can enter 0 at any time to go back to the main menu, and enter 0 to exit from the application (Figure 16-13).



```
*************************************************


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 16-13 Exit

16.7 Summary

This component of the scenario demonstrates enhanced features of the WebSphere MQ V7.0 implementation of Java Message Service.

The store ordering application uses the wildcard (+) to subscribe. This allows the application to fetch only the immediate child topics.

The retail order application manages retained publications stored by the WebSphere MQ queue manager at headquarters.

The connection factory objects can be configured to make use of the asychcronous put and the read ahead features.

17

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 detailed in Chapter 5, “WebSphere MQ Client enhancements” on page 59.

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 43, for details on Publish/Subscribe and the new topic object.

This chapter consists of the following sections:

17.1 Design overview

News items are generated at the headquarters of Matt's Deli and they are available to be viewed by the store staff and the general public. This can be news like internal staff bulletins, announcements of special offers, or new products. Each news item is stored as a separate MQ message in plain text.

Figure 17-1 shows the overall design. The programs and data are independent of all other components of the scenario.

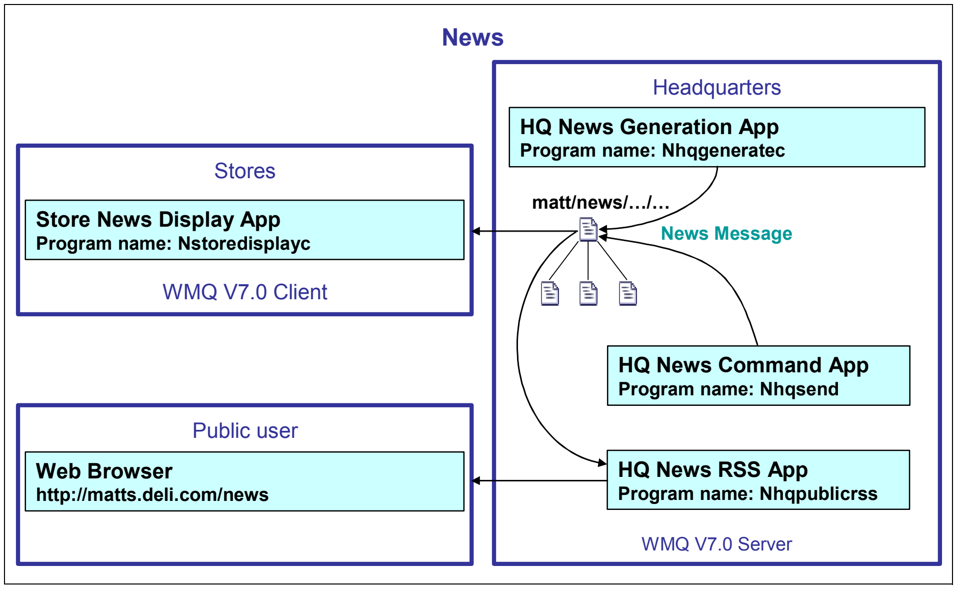


Figure 17-1 Programs and data flows in the news component

News items are published to two layers of topics beneath the tree matt/news on the headquarters queue manager using two different programs that are run by headquarters staff. One of these is designed to read news from a text file that 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 soon 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. A RSS feed is also available in this scenario. A Web browser with an appropriate RSS reader plug-in can view the latest news on all the public topics, or a dedicated RSS reader program can be used.

There is a JMS program that runs continuously at headquarters. It subscribes to all public news topics and accumulates them in an XML file in RSS-compliant format.

17.2 Deploying the news component

Some of the programs are designed to be run on the same system as the headquarters queue manager. The programs at the stores and public Web browser 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 useful if the programs are to be run on UNIX and other platforms that support C.

Listings of the data files and Windows command files are provided when they are used in 17.4, “Verifying the news component” on page 334.

17.2.1 Copying files

The file Chapter17_News.zip is supplied with the materials in Appendix B, “Additional material” on page 379. To deploy the news component, extract the compressed file into the C:\Scenario folder on the headquarters and stores systems. This creates a sub-folder called Chapter17_News. Table 17-1 contains a list of the files required to run the component.

Table 17-1 News component fil

Program	File names	Description
News generation	Nhqgeneratec.exe	Windows executable program that runs at headquarters. It uses the 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.
	testgendata1.txt	Sample data for generating news items on topics.
	runnewsgendelay.bat runnewsgenfast.bat	Command files to run the Nhqgeneratec program with specific parameters.

Program	File names	Description
News command	Nhqsend.exe	Windows executable program that runs at headquarters. It reads data records from standard input and publishes 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 that 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 that 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.
	runstoredisplay.bat	Command file to run the Nstoredisplayc program with specific parameters.

17.2.2 Public Web browser

A Web browser, such as Microsoft Internet Explorer or Mozilla Firefox with an 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. A dedicated RSS reader program can also be used. These are not supplied with the materials in Appendix B, “Additional material” on page 379.

17.2.3 Compiling the Windows executable programs

The C source files for all the programs are provided in the file Chapter17_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 379.

The make file must be changed to allow the programs to be compiled and linked on UNIX. Details of the required changes are not covered in this book, as they are dependent on the UNIX platform. For details on compilation of a C program, refer to the *Application Programming Guide* manual, available in the WebSphere MQ V7.0 Information Center at:

`http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`

The files are listed in Table 17-2.

Table 17-2 Files used to compile the news component

File names	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 that uses makenews.txt to compile and link the C programs on Windows.

17.2.4 MQ Objects

The news component uses the queue manager at headquarters and does not require any queue objects to be defined. It is purely Publish/Subscribe based, and lets WebSphere 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 channel objects to be defined so that MQ Client programs can connect to the queue manager from a remote system and the headquarters system.

The MQSC commands in Example 17-1 are included in QMHQobjects.txt, which is run during the scenario preparation steps described in Chapter 14, “Scenario preparation” on page 269.

Example 17-1 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 Object Authority Manager (OAM) commands in Example 17-2 are included in QMHQsetaut.bat, which is also run during the scenario preparation.

Example 17-2 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.

17.2.5 RSS data directory

Create the directory C:\htmldata on the headquarters system. The Nhqpublicrss Java program writes RSS XML data to the file news.xml in this directory.

17.3 Running the News component

In a Publish/Subscribe design using non-durable subscriptions and non-retained publications, it is typical to start the subscribers first and then run the publishers.

Messages for non-retained publications 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 about how to run the programs, including details of the environment and command-line arguments that they require.

17.3.1 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 sam725hq and port number 1414 must be changed to whatever is actually being used by the headquarters queue manager:

```
set MQSERVER=QMHQ_STORES/TCP/sam725hq(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 Table 17-3.

Table 17-3 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
```

17.3.2 Starting the news RSS generation program at headquarters

Nhqpublicrss runs on the headquarters system. It is a compiled Java class that subscribes to the public news topics and generates a RSS XML file on C:\htmldata\news.xml. 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
```

17.3.3 Starting the public news RSS reader

Open the following URL in a Web browser that has a RSS plug-in or open it in a special-purpose RSS reader program:

```
http://matts.deli.com/htmldata/news.xml
```

matts.deli.com needs to be changed to the host name of the headquarters system. It displays the public news items.

17.3.4 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.

In a Windows command prompt, set the MQSERVER variable. The host sam725hq and port number 1414 must be changed to whatever is actually being used by the headquarters queue manager:

```
set MQSERVER=QMHQ_NEWS/TCP/sam725hq(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 17-3 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 to repeat the set of news items a specified number of times. This does not make best use of the asynchronous put feature, but it does simulate a regular feed of news items that can be subscribed to and displayed by the other programs:

```
Nhqgeneratec -delay 5 -repeat 100 <testgendata1.txt
```

This introduces a 5-second delay between each published news item and repeats all the news items in the data file 100 times before the program termination.

17.3.5 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. See Table 17-4.

Table 17-4 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
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.

17.4 Verifying the news component

This section provides two test cases that 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.

17.4.1 News is generated at headquarters and displayed by stores

On a store system:

1. Start the program to subscribe to and display all news topics by double-clicking **runstoredisplay.bat** in Windows Explorer. The file contains the following commands. The host sam725hq and port number 1414 must be changed to whatever is actually being used by the headquarters queue manager.

   ```
   set MQSERVER=QMHQ_STORES/TCP/sam725hq(1414)
   Nstoredisplayc all
   pause
   ```

The program displays news items for all the internal and public topics. The command prompt window should display the output shown in Example 17-4.

Example 17-4 Initial output of news display program at store

```
C:\Scenario\Chapter17_News> set MQSERVER=QMHQ_STORES/TCP/sam725hq(1414)
C:\Scenario\Chapter17_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.

2. On the headquarters system, start the program to quickly publish test data to all five news topics by double-clicking **runnewsgenfast.bat** in Windows Explorer. The file contains the following commands:

   ```
   set MQSERVER=QMHQ_NEWS/TCP/localhost(1414)
   Nhqgeneratec -fast <testgendata1.txt
   pause
   ```

   The command prompt window should display the output shown in Example 17-5.

Example 17-5 Output of news generation program at headquarters

```
C:\Scenario\Chapter17_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\Chapter17_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 in Example 17-6.

Example 17-6 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
```

3. On the headquarters system, start the program to interactively publish news items to one specific topic by double-clicking **runnewssend.bat** in Windows Explorer. The file contains the following commands:

   ```
   Nhqsend internal/staff
   pause
   ```

4. 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 Example 17-7. The user input is shown in bold.

Example 17-7 Output of news send program at headquarters

```
C:\Scenario\Chapter17_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\Chapter17_News> pause
Press any key to continue . . .
```

5. Press any key to close the window. At the same time, the news display program window should have progressively displayed the output in Example 17-8.

Example 17-8 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\Chapter17_News> pause
Press any key to continue . . .
```

Press any key to close the window. This concludes the verification of news generation at headquarters and news display at the stores.

The next section runs the RSS generation program and verifies that a RSS reader can view the news items.

17.4.2 News is generated and displayed by a RSS reader

On the headquarters system, start the Java program to subscribe to all public news topics and generate the RSS XML file. A Windows command file is provided to compile and run the program. The files contain the commands listed in Example 17-9.

Example 17-9 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 **runrssgenxml.bat** in Windows Explorer. Press any key when prompted. The command prompt window should initially display the output shown in Example 17-10.

Example 17-10 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 **runnewsgendelay.bat** in Windows Explorer. The file contains the following commands:

```
set MQSERVER=QMHQ_NEWS/TCP/localhost(1414)
Nhqgeneratec -delay 2 -repeat 10 <testgendata1.txt
pause
```

The program reads all the records from standard input into memory and then publishes news items with a 2-second delay between each. It repeats the sequence of records 10 times before terminating.

After about 15 seconds of running the command prompt window, the output obtained is as shown in Example 17-11.

Example 17-11 Initial output of news generation program at headquarters

```
C:\Scenario\Chapter17_News> set MQSERVER=QMHQ_NEWS/TCP/localhost(1414)
C:\Scenario\Chapter17_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 that has a RSS plug-in, or open it in a special purpose RSS Reader program:

```
http://matts.deli.com/htmldata/news.xml
```

matts.deli.com must be changed to the actual 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 shown in Example 17-12.

Example 17-12 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\Chapter17_News> pause
Press any key to continue . . .
```

17.5 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. The 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 acknowledged the success of all MQPUTs.

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 WebSphere MQ V7.0 compared to the previous versions of WebSphere MQ.

18

Scenario: Web ordering over HTTP

This component of the scenario illustrates the WebSphere MQ Bridge for HTTP that is supplied with WebSphere MQ V7.0. The bridge is deployed in an Application Server, allowing a Web client application to directly interact with WebSphere MQ without any MQ code installed on the client machine.

Both Publish/Subscribe and point-to-point messaging are supported by the bridge, but this scenario component only demonstrates a simple request and reply dialog using point-to-point. The JavaScript programming language is used within a HTML page that is run in a Web browser.

This chapter includes the following sections:

18.1 Design overview

Matt’s Deli has recently introduced the ability for customers to order products via the Internet. Customers use a Web page to place orders that are processed by Matt’s Deli headquarters. The warehouse arranges delivery.

The Web ordering component uses the same WebSphere MQ back-end application as the store ordering component. The back-end retail order application runs continuously at headquarters to process retail orders from stores and the Web.

Customers open the Matt’s Deli Web page and submit three pieces of information to place an order:

- The catalog ID of the product
- The quantity that they require
- Their customer ID

The JavaScript Web ordering application in the Web page generates an HTTP POST request to place a message on queue MATT.RETAIL.ORDERS. The message on this queue is processed by the back-end retail order application at headquarters and generates a response message on queue MATT.RETAIL.RESPONSES.

The Web ordering application performs a HTTP DELETE request to receive the response message for the order and then displays it to the customer. If the headquarters application does not generate a response message within 5 seconds the Web ordering application displays an error message.

The customer can then place a new order if they wish.

Figure 18-1 depicts the workflow of the entire Web ordering component.

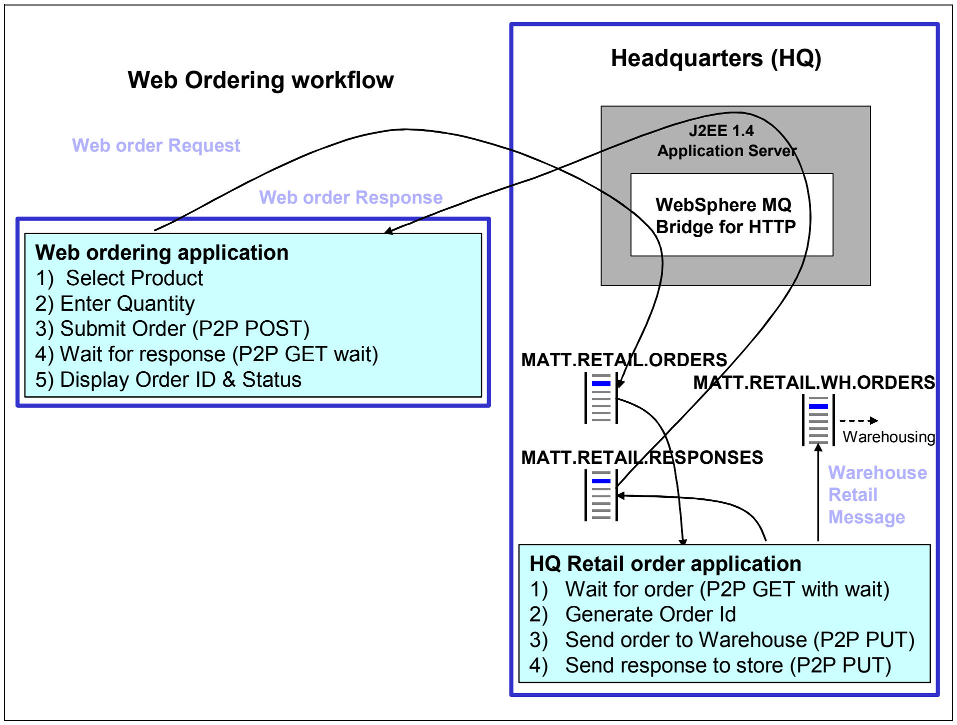


Figure 18-1 Web ordering component

HTTP requests are processed by the WebSphere MQ Bridge for HTTP, which is deployed as a servlet in the WebSphere Application Server that is running on the headquarters machine.

Web ordering is a zero footprint application because the JavaScript code does not require WebSphere MQ code to be installed on customer machines.

18.2 Prerequisites

This section describes software products and applications that are prerequisites to running the WebSphere MQ Bridge for HTTP and the Web ordering component.

18.2.1 WebSphere Application Server on headquarters machine

WebSphere Application Server V6.1.0.0 must be installed on the headquarters machine on which the headquarters WebSphere MQ V7.0 queue manager also resides.

For instructions to install WebSphere Application Server, refer to the installation guide available at:

http://www.ibm.com/software/webservers/appserv/was/library/

The default profile for WebSphere Application Server is used in this scenario, as shown in Example 18-1.

Example 18-1 Profile for 'server1'

```
Server name is:server1
Profile name is:AppSrv01
Profile home is:C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01
Profile type is:default
Cell name is:WMQV7GANode01Cell
Node name is:WMQV7GANode01
Current encoding is:Cp1252
Server port number is:9080
```

Start WebSphere Application Server and ensure that server1 is running. This can be done by opening a command prompt window and entering the commands shown in Example 18-2.

Example 18-2 Starting WebSphere Application Server

```
C:\>cd "C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin"
C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin>startServer
server1
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\startServer.log
ADMU7701I: Because server1 is registered to run as a Windows Service, the
           request to start this server will be completed by starting the
           associated Windows Service.
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\startServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 4771
```

Note: A WebSphere MQ resource adapter is required for connecting other application servers to WebSphere MQ V7.0. For example, WebSphere Application Server Community Edition (WAS CE) can be used in conjunction with the WebSphere MQ V7.0 JCA Resource Adapter to connect to WebSphere MQ V7.0. A description of deploying the scenario in other application servers is beyond the scope of this book because every application server has a different procedure.

18.2.2 Retail order application on headquarters machine

Deploy the headquarters retail order application, as used by the store ordering component of the scenario. Refer to 16.2, “Deploying the headquarters application” on page 305, for instructions.

18.2.3 Web browser on client machine

The client machine that is used by a Web customer to make orders requires a Web browser that supports JavaScript, such as Internet Explorer or Mozilla Firefox. No other software is needed on this machine.

18.3 Deploying the Web ordering component

The WebSphere MQ Bridge for HTTP is a Java EE servlet and it can be deployed within the servlet container of any Java EE 1.4 compliant application server. WebSphere Application Server V6.1.0.0 is used in this component of the scenario.

This section describes the configuration of WebSphere Application Server and the deployment of the bridge and the Web ordering application.

18.3.1 Start WebSphere Application Server administrative console

Launch the WebSphere Application Server administrative console on the headquarters system by using the start menu and selecting **Programs** → **IBM WebSphere** → **Application Server V6.1** → **Profiles** → **AppSrv01** → **Administrative console**.

An alternative is to open the URL `http://localhost:9060/ibm/console/` in a Web browser on the headquarters system.

18.3.2 Creating a WebSphere MQ connection factory

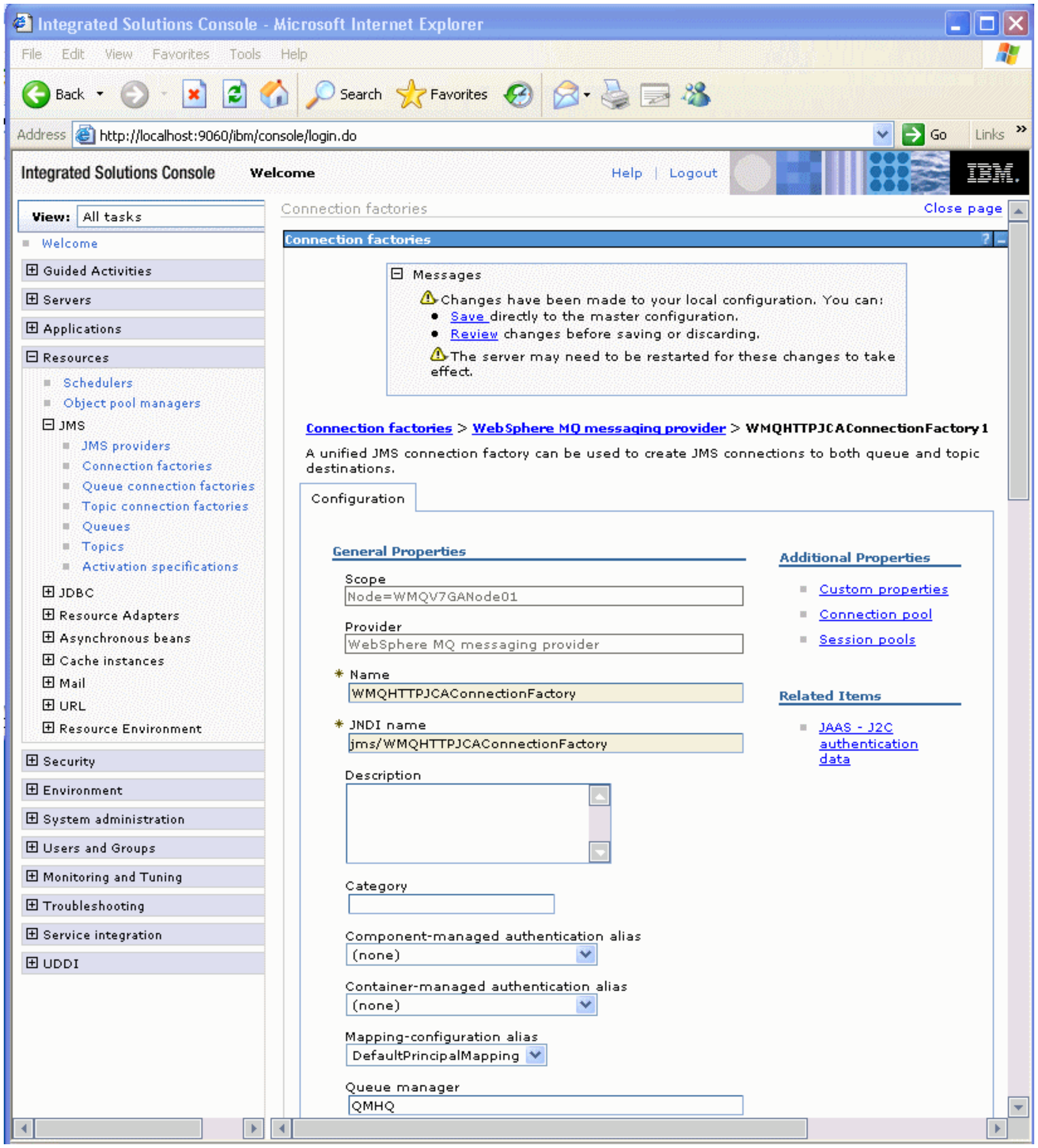
In the administrative console, navigate through **Resources** → **JMS** → **Connection factories**. Set the scope to **WMQV7GANode01** → **New**. Select **WebSphere MQ messaging provider** → **OK**.

Enter the properties for the connection factory as provided Example 18-3.

Example 18-3 Connection factory properties

```
Name = WMQHTTPJCAConnectionFactory
JNDI Name = jms/WMQHTTPJCAConnectionFactory
Queue manager = QMHQ
Transport Type = Bindings
```

Figure 18-2 shows the connection factory creation. Select **OK** and save the connection factory.



Integrated Solutions Console - Microsoft Internet Explorer

Address http://localhost:9060/ibm/console/login.do

Integrated Solutions Console Welcome Help | Logout

View: All tasks

Connection factories

Messages

Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

Connection factories > WebSphere MQ messaging provider > WMQHTTPJCAConnectionFactory1

A unified JMS connection factory can be used to create JMS connections to both queue and topic destinations.

Configuration

General Properties

Scope: Node=WMQV7GANode01

Provider: WebSphere MQ messaging provider

* Name: WMQHTTPJCAConnectionFactory

* JNDI name: jms/WMQHTTPJCAConnectionFactory

Description:

Category:

Component-managed authentication alias: (none)

Container-managed authentication alias: (none)

Mapping-configuration alias: DefaultPrincipalMapping

Queue manager: QMHQ

Additional Properties

- Custom properties
- Connection pool
- Session pools

Related Items

- JAAS - J2C authentication data

Figure 18-2 Create WebSphere MQ connection factory

Restart WebSphere Application Server. This can be done by opening a command prompt window and entering the commands shown in Example 18-4.

Example 18-4 Restarting WebSphere Application Server

```
C:\>cd "C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin"
C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin>stopServer
server1
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\stopServer.log
ADMU7702I: Because server1 is registered to run as a Windows Service, the
           request to stop this server will be completed by stopping the
           associated Windows Service.
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\stopServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.

C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin>startServer
server1
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\startServer.log
ADMU7701I: Because server1 is registered to run as a Windows Service, the
           request to start this server will be completed by starting the
           associated Windows Service.
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\startServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 4776
```

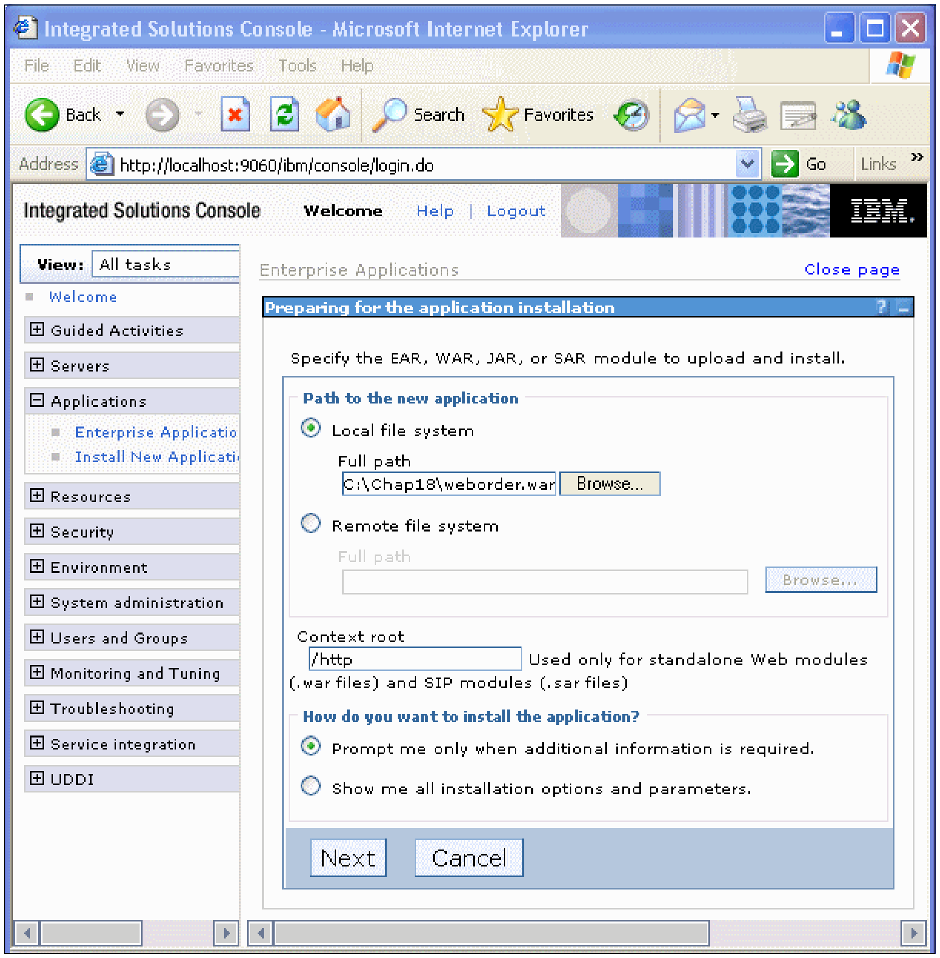
18.3.3 Installing the Web ordering servlet

The Chapter18_WebOrdering.zip file is supplied with the materials in Appendix B, "Additional material" on page 379. Extract the compressed file into the C:\Scenario directory. This creates the Chapter18_WebOrdering sub-directory, which contains the file weborder.war.

In the WebSphere Application Server Administrative Console:

1. Navigate to **Applications** → **Enterprise Applications** and select **Install**.
2. Browse to the file weborder.war in the C:\Scenario\Chapter18_WebOrdering directory.

3. Set the context root to /http, as shown in Figure 18-3. This distinguishes the application from other applications already deployed in the WebSphere Application Server.


Integrated Solutions Console - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:9060/ibm/console/login.do Go Links
Integrated Solutions Console Welcome Help | Logout
View: All tasks
Welcome
Guided Activities
Servers
Applications
Enterprise Applicatio
Install New Applicati
Resources
Security
Environment
System administration
Users and Groups
Monitoring and Tuning
Troubleshooting
Service integration
UDDI
Enterprise Applications Close page
Preparing for the application installation

Specify the EAR, WAR, JAR, or SAR module to upload and install.

Path to the new application

[✓ Local file system]

Full path: C:\Chap18\weborder.war Browse...

Remote file system

Full path Browse...

Context root: /http Used only for standalone Web modules (.war files) and SIP modules (.sar files)

How do you want to install the application?

[✓ Prompt me only when additional information is required.]

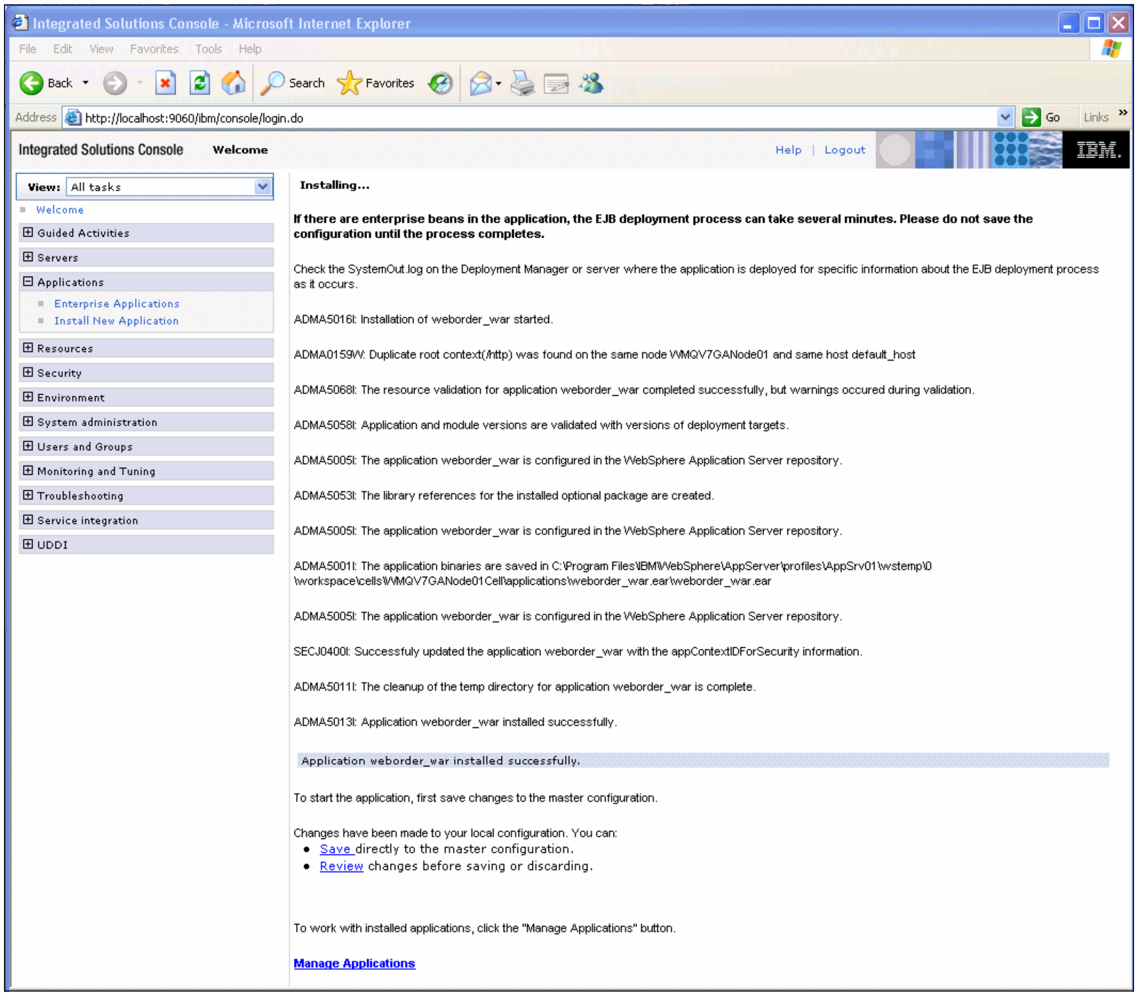
Show me all installation options and parameters.

Next Cancel

Figure 18-3 Installing servlet

4. Select **Next** for subsequent option pages, leaving the default options present, to finally select **Finish** in the summary page.

5. On successful installation of the servlet, select **Save to Master Configuration**, as shown in Figure 18-4.



Installing...

If there are enterprise beans in the application, the EJB deployment process can take several minutes. Please do not save the configuration until the process completes.

Check the SystemOut.log on the Deployment Manager or server where the application is deployed for specific information about the EJB deployment process as it occurs.

```
ADMA5016I: Installation of weborder_war started.

ADMA0159W: Duplicate root context(/http) was found on the same node WMQV7GANode01 and same host default_host

ADMA5068I: The resource validation for application weborder_war completed successfully, but warnings occured during validation.

ADMA5058I: Application and module versions are validated with versions of deployment targets.

ADMA5005I: The application weborder_war is configured in the WebSphere Application Server repository.

ADMA5053I: The library references for the installed optional package are created.

ADMA5005I: The application weborder_war is configured in the WebSphere Application Server repository.

ADMA5001I: The application binaries are saved in C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\wstemp\0
\workspace\cells\WMQV7GANode01Cell\applications\weborder_war.ear\weborder_war.ear

ADMA5005I: The application weborder_war is configured in the WebSphere Application Server repository.

SECJ0400I: Successfuly updated the application weborder_war with the appContextIDForSecurity information.

ADMA5011I: The cleanup of the temp directory for application weborder_war is complete.

ADMA5013I: Application weborder_war installed successfully.
```

Application weborder_war installed successfully.

To start the application, first save changes to the master configuration.

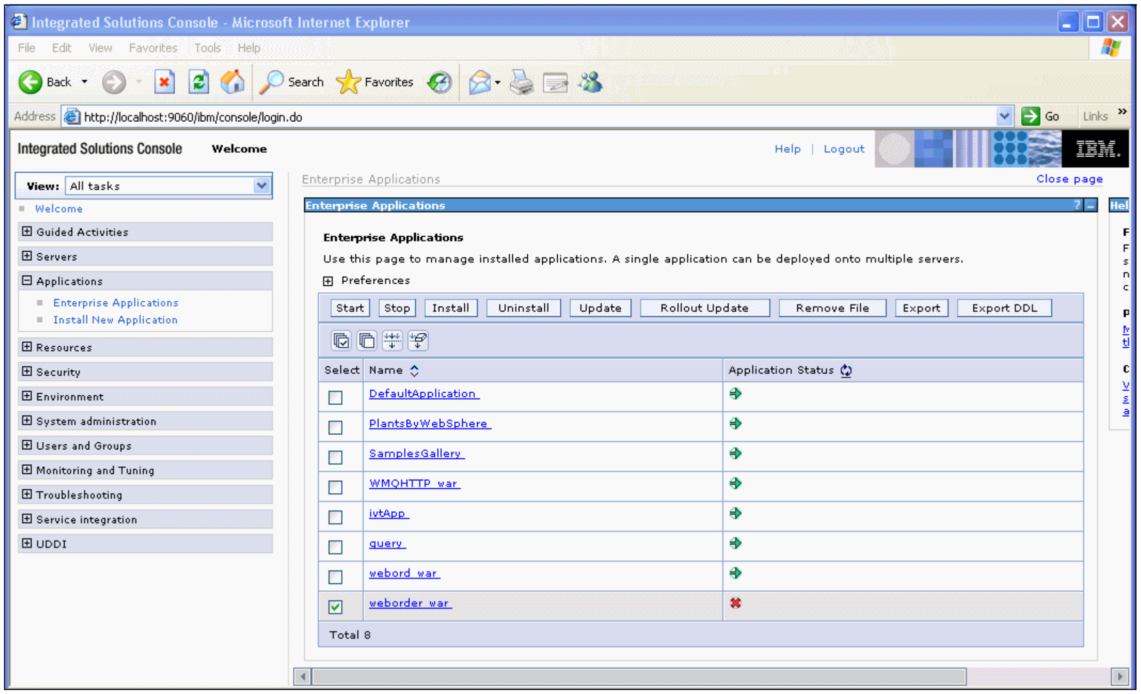
Changes have been made to your local configuration. You can:
- Save directly to the master configuration.
- Review changes before saving or discarding.

To work with installed applications, click the "Manage Applications" button.

Manage Applications

Figure 18-4 Installing and saving the servlet to master configuration

6. After successful installation of the servlet, navigate to the Enterprise Application page and start the application, as shown in Figure 18-5.


Integrated Solutions Console - Microsoft Internet Explorer

Address http://localhost:9060/ibm/console/login.do

Integrated Solutions Console Welcome Help | Logout

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
  - Enterprise Applications
  - Install New Application
- Resources
- Security
- Environment
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
- UDDI

Enterprise Applications

Enterprise Applications

Use this page to manage installed applications. A single application can be deployed onto multiple servers.

Preferences

Start | Stop | Install | Uninstall | Update | Rollout Update | Remove File | Export | Export DDL

Select	Name	Application Status
☐	DefaultApplication	➔
☐	PlantsByWebSphere	➔
☐	SamplesGallery	➔
☐	WMQHTTP_war	➔
☐	ivtApp	➔
☐	query	➔
☐	webord_war	➔
☑	weborder_war	✖

Total 8

Figure 18-5 Starting the application after installation and deployment

Note: The deployment and installation can be verified using the following URL in a Web browser on the headquarters machine:

```
http://localhost:9080/mq/msg/queue/SYSTEM.DEFAULT.LOCAL.QUEUE
```

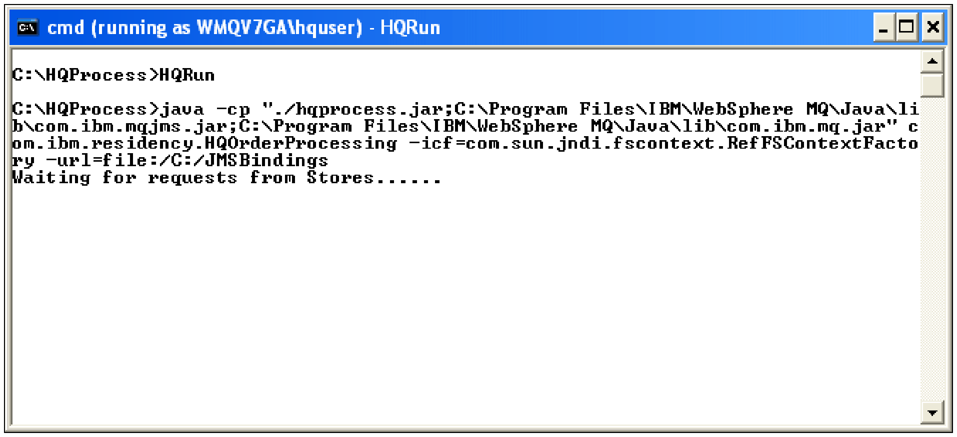
If the WebSphere MQ Bridge for HTTP is correctly installed a MQ error message is displayed in the Web browser. This confirms that the WebSphere MQ Bridge for HTTP has successfully connected to the WebSphere MQ queue manager.

18.4 Running the Web ordering component

The bridge connects via JMS to the headquarters queue manager. This allows the Web ordering application to communicate with the store ordering application. The Web ordering application uses a simple point-to-point interface to send a product order message to headquarters by performing a HTTP POST request and receives the order confirmation message by performing a HTTP DELETE request. It uses the same retail product order message protocol and queues as the store ordering component described in Chapter 16, “Scenario: Store ordering with JMS” on page 303.

18.4.1 Starting the headquarters retail order application

Invoke the retail order application as described in 16.3, “Running the headquarters application” on page 308. This runs continuously at headquarters to process orders from stores and Web customers. The initial execution is shown in Figure 18-6.

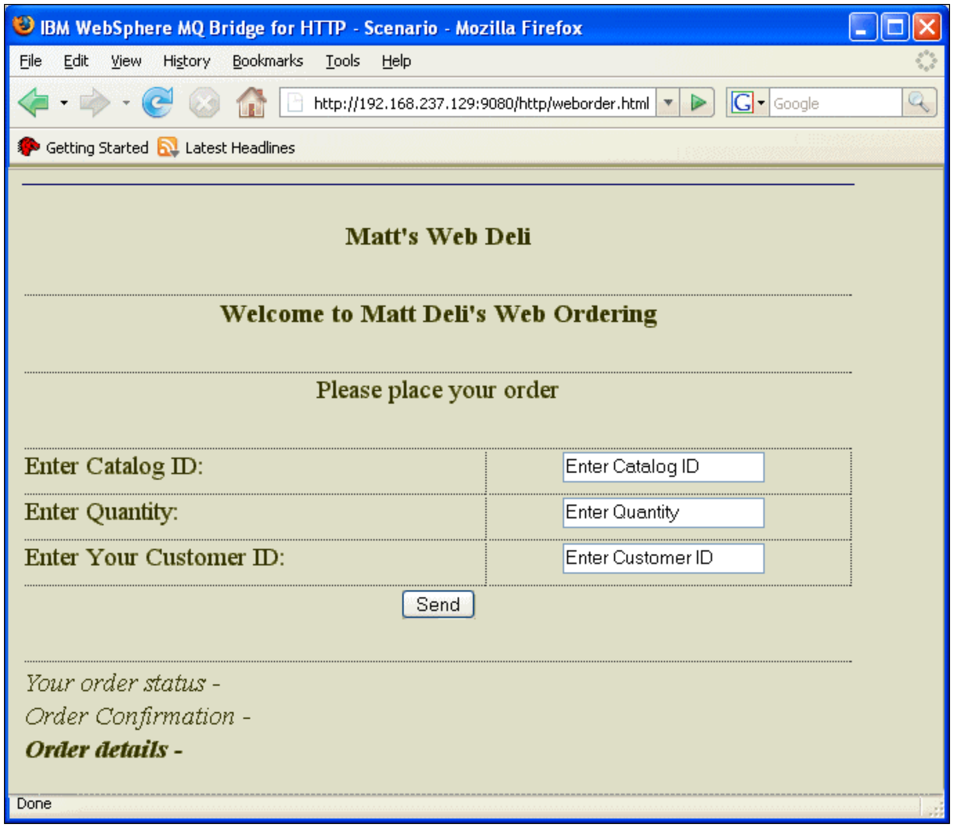


```
C:\HQProcess>HQRun

C:\HQProcess>java -cp "./hqprocess.jar;C:\Program Files\IBM\WebSphere MQ\Java\li
b\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.jar" c
om.ibm.residency.HQOrderProcessing -icf=com.sun.jndi.fscontext.RefFSContextFacto
ry -url=file:/C:/JMSBindings
Waiting for requests from Stores......
```

Figure 18-6 Starting the headquarters application

18.4.2 Invoking the Web ordering application from the Web page

Invoke the Web ordering application in a Web browser, as shown in Figure 18-7, using the URL `http://www.matts.deli.com/http/weborder.html`, which is of the format `http://HostName:9080/http/weborder.html`.



IBM WebSphere MQ Bridge for HTTP - Scenario - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.237.129:9080/http/weborder.html
Google
Getting Started
Latest Headlines
Send
Done

Matt's Web Deli

Welcome to Matt Deli's Web Ordering

Please place your order

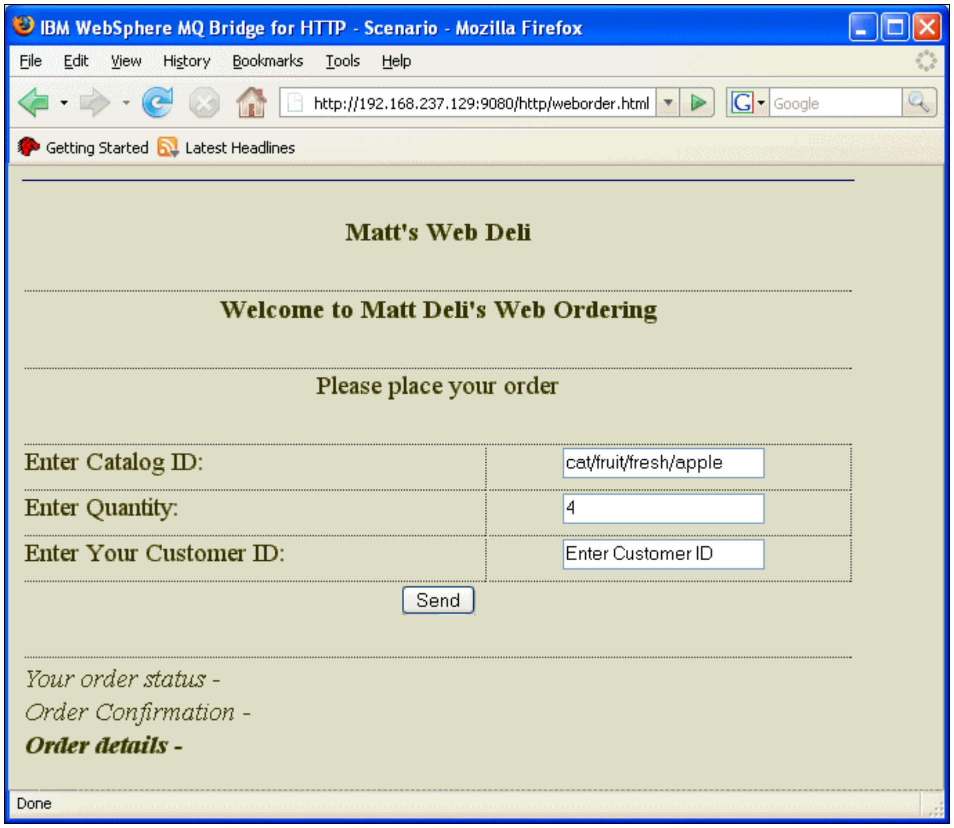
Enter Catalog ID:	Enter Catalog ID
Enter Quantity:	Enter Quantity
Enter Your Customer ID:	Enter Customer ID

Your order status -
Order Confirmation -
Order details -

Figure 18-7 Web ordering application invocation

18.4.3 Component verification

Enter the catalog ID for the required order, followed by the Quantity and Customer ID fields, as shown in Figure 18-8.


IBM WebSphere MQ Bridge for HTTP - Scenario - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.237.129:9080/http/weborder.html
Google
Getting Started
Latest Headlines
Send
Done

Matt's Web Deli

Welcome to Matt Deli's Web Ordering

Please place your order

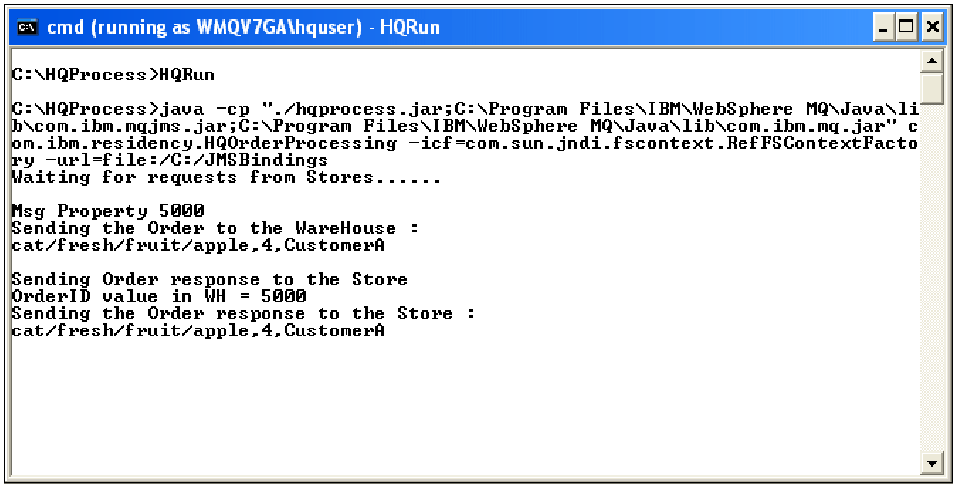
Enter Catalog ID:	cat/fruit/fresh/apple
Enter Quantity:	4
Enter Your Customer ID:	Enter Customer ID

Your order status -
Order Confirmation -
Order details -

Figure 18-8 Placing a Web order

Submit the order by clicking **Send**.

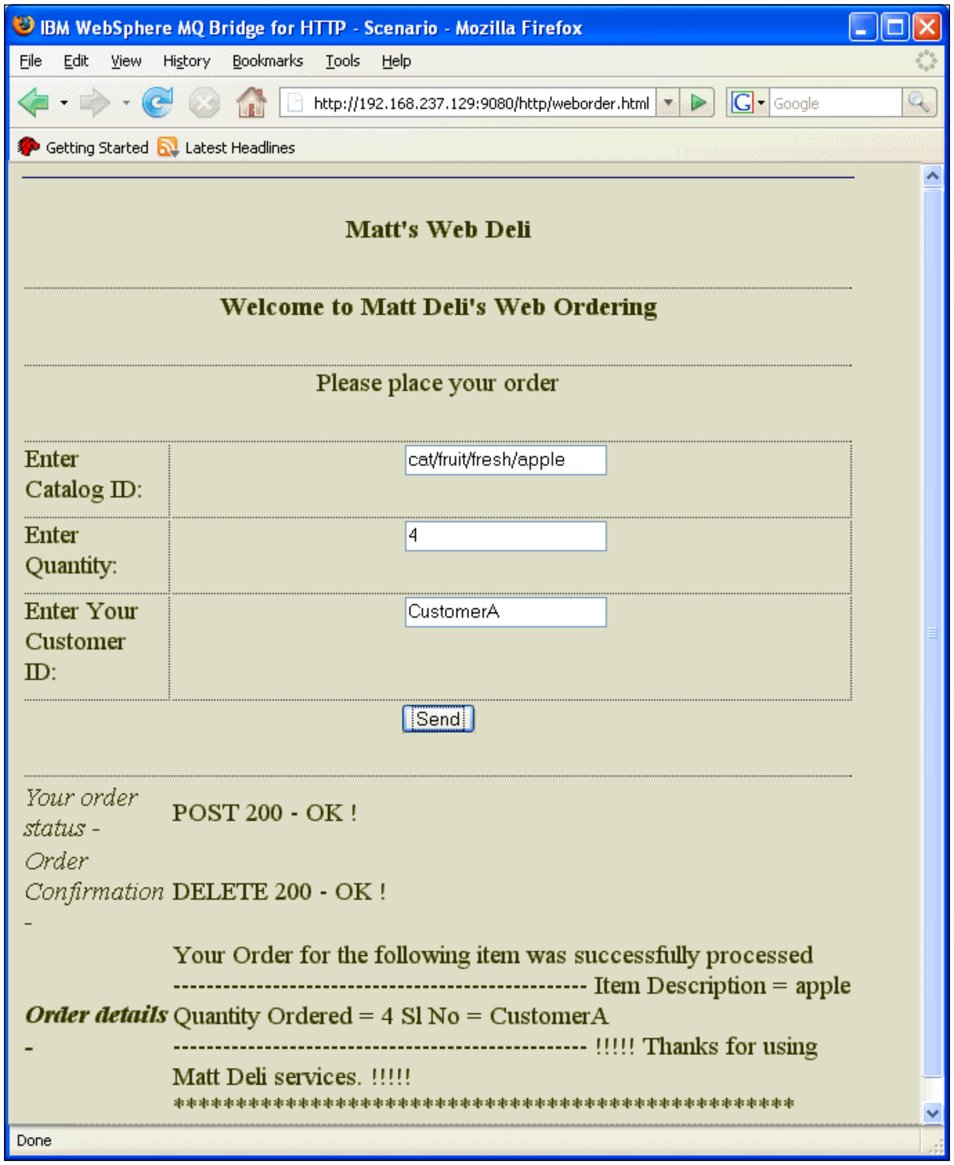
The Retail order application running at the headquarters then processes the request and sends a response back to the client, as shown in Figure 18-9.



```
C:\HQProcess>HQRun

C:\HQProcess>java -cp "./hqprocess.jar;C:\Program Files\IBM\WebSphere MQ\Java\li
b\com.ibm.mqjms.jar;C:\Program Files\IBM\WebSphere MQ\Java\lib\com.ibm.mq.jar" c
om.ibm.residency.HQOrderProcessing -icf=com.sun.jndi.fscontext.RefFSContextFacto
ry -url=file:/C:/JMSBindings
Waiting for requests from Stores......

Msg Property 5000
Sending the Order to the WareHouse :
cat/fresh/fruit/apple,4,CustomerA

Sending Order response to the Store
OrderID value in WH = 5000
Sending the Order response to the Store :
cat/fresh/fruit/apple,4,CustomerA
```

Figure 18-9 Headquarters application having processed the order

The response obtained on successful order processing is shown in Figure 18-10.


IBM WebSphere MQ Bridge for HTTP - Scenario - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.237.129:9080/http/weborder.html
Google
Getting Started Latest Headlines
Done

Matt's Web Deli

Welcome to Matt Deli's Web Ordering

Please place your order

Enter Catalog ID:	cat/fruit/fresh/apple
Enter Quantity:	4
Enter Your Customer ID:	CustomerA

Send

Your order status - POST 200 - OK !

Order Confirmation - DELETE 200 - OK !

Order details - Your Order for the following item was successfully processed -- Item Description = apple Quantity Ordered = 4 Sl No = CustomerA -- !!!!! Thanks for using Matt Deli services. !!!!! **

Figure 18-10 Web order response

18.5 Summary

This component of the scenario demonstrates the WebSphere MQ Bridge for HTTP that is supplied with WebSphere MQ V7.0.

The Web ordering application is a zero footprint client, with no WebSphere MQ libraries installed on the client machine. The application is a servlet that is hosted by WebSphere Application Server. It uses HTTP requests to implement a simple point-to-point WebSphere MQ interface to send a request message and receive a response message.

19

Scenario: Warehousing using call back

This component of our scenario illustrates the use of callback for asynchronous consumption of WebSphere MQ messages. It also demonstrates how an administrative subscription can be setup to send published messages to a remote queue. The messages are processed by a point-to-point application that is not aware of Publish/Subscribe.

This chapter contains the following sections:

19.1 Design overview

Matt’s Deli currently has one warehouse that receives deliveries of product stocks from all suppliers, provides storage, and dispatches products to the stores and Web customers in response to orders received. The warehouse management system runs on a z/OS mainframe system in the warehouse data center. All orders are received via WebSphere MQ messages from the headquarters system.

The warehouse z/OS mainframe system has recently upgraded its queue manager to WebSphere MQ V7.0 for z/OS. The warehouse applications were not designed to use Publish/Subscribe. They rely on a point-to-point design for application integration and communication.

The warehouse has modified its main application to use the new callback feature in WebSphere MQ V7.0, enabling MQ to allocate message buffers with the appropriate length to receive the order messages. This is very useful because the messages can vary in size from a hundred bytes for simple orders from Web customers up to many megabytes for large bulk orders from stores. The application has also been modified to display message properties using new MQI calls provided in WebSphere MQ V7.0.

The warehousing component of this scenario consists of two application programs:

- A gateway application runs on the headquarters system to process orders that are accumulated on the MATT.RETAIL.WH.ORDERS queue. It publishes the orders to topics matt/warehouse/*CatalogId*. 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 strategy. Pub/Sub enables new warehouses to subscribe to appropriate topics and receive orders that they can fulfill, based on the type of stock that they have.
- An application runs on the warehouse z/OS system to receive orders. It gets messages from the local queue MATT.WH.ORDERS and processes them. In this scenario it merely displays them on SYSOUT. This application implements callback for asynchronous consume.

Figure 19-1 shows the two application programs and the interaction between headquarters and the warehouse.

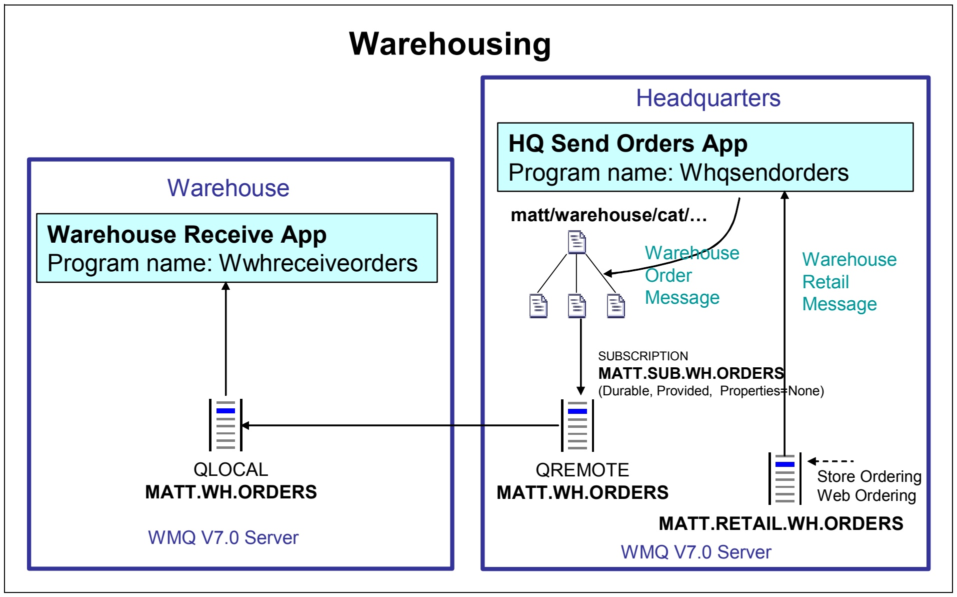


Figure 19-1 Warehousing scenario component

19.2 Deploying the warehousing component

The warehousing component is made of two application programs. One program runs on the headquarters Windows system and the other runs on the warehouse z/OS system:

- HQ send orders (Whqsendorders) runs on the headquarters system.

  This is a C language program that gets messages from the MATT.RETAIL.WH.ORDERS queue and publishes them to topics matt/warehouse/*CatalogId*.

- Warehouse receive orders (Wwhreceiveorders) runs on the warehouse z/OS system as a batch program.

  This program is provided in two languages, C and COBOL. The program gets messages from local queue MATT.WH.ORDERS using a callback consumer function. Orders are displayed on the standard output of the program (SYSOUT).

  The C program runs until it is cancelled by the operator or a message is received with the text '`shutdown`' in the message body.

The COBOL program runs until the operator responds with the word “QUIT” to the system console message “TSTMQCBM Enter START, STOP or QUIT”.

This component requires the creation of administrative subscription MATT.SUB.WH.ORDERS on the headquarters queue manager to topic matt/warehouse/#. This 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 queue manager QMWH running in the warehouse. This subscription converts publications made on the headquarters queue manager into messages on a local queue in the warehouse queue manager.

Sender and receiver channels are configured between queue managers QMHQ and QMWH to transport the messages to the warehouse.

19.2.1 Required MQ objects

The warehousing component requires the following objects to be defined in the headquarters QMHQ queue manager. Example 19-1 shows the MQSC commands to define them.

- MATT.TOPIC.WAREHOUSE: This topic object equates to the topic string matt/warehouse. This topic object is used by the headquarters 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 (non-managed) subscriber queue is MATT.WH.ORDERS, which 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 16, “Scenario: Store ordering with JMS” on page 303.

Example 19-1 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 MATT.WH.ORDERS object is required on the warehouse QMWH queue manager. Example 19-2 contains the MQSC command to define it. It is local queue that receives the publications that match the subscription MATT.SUB.WH.ORDERS.

Example 19-2 Warehousing queue for the z/OS queue manager

```
DEFINE QLOCAL(MATT.WH.ORDERS) +
   DEFPSIST(YES) +
   DEFSOPT(SHARED) +
   REPLACE
```

Transmission queues and sender and receiver channels are required to interconnect the QMHQ and QMWH queue managers.

19.2.2 Installation of warehousing component

The warehousing programs are packaged in a compressed file called Chapter19_Warehousing.zip that is supplied with the materials in Appendix B, “Additional material” on page 379. It contains source code, executable code, and the make file to build the C language executables that run at headquarters. The file also includes the source code and sample Job Control Language (JCL) to compile and link the C program or COBOL program that runs on the warehouse z/OS system.

This file also has a compiled copy of the C language program for receiving orders (Wwhreceiveorders.exe) that can run on a Windows system, instead of requiring a z/OS system to operate the warehouse application.

To install the executable code on Windows:

1. Extract the contents of the compressed file into the directory C:\Scenario.
2. The following directories are created by the extraction:
   - C:\Scenario\Chapter19_Warehousing\windows
   - C:\Scenario\Chapter19_Warehousing\zos
3. Change to the C:\Scenario\Chapter19_Warehousing\windows directory.
4. This directory contains files using the following naming convention:
   - *ProgramName*.c: This is the program source code.
   - *ProgramName*.exe: This is the executable code linked with the MQ server bindings library.
   - make.bat: This is 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.
   - test*ProgramName*.bat: This is a command file to execute the program on a Windows system.

These programs are compiled with a C language compiler such as the free *Microsoft Visual C++ Express Edition* for Windows.

The following instructions describe how to install the program on the warehouse z/OS system. It assumes that the reader is familiar with both Windows and z/OS platforms:

1. Change to the C:\Scenario\Chapter19_Warehousing\zos\C directory.
2. This directory contains the following files for the C program:
   - 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.

3. Change to the C:\Scenario\Chapter19_Warehousing\zos\COBOL directory. This directory contains the following files for the COBOL program:
   - TSTMQCBM.cbl: This is the source code of the receive orders *main* program in COBOL language.
   - TSTMQCBF.cbl: This is the source code of the receive orders callback function program in COBOL language.
   - MQCOBOL.jcl: This is a sample JCL to compile and link the COBOL language programs in z/OS.
   - TSTMQCB.jcl: This is a sample JCL to run the TSTMQCBM program.
4. On the z/OS system, create PDS libraries to store the JCL and the source code.
5. Transfer the source code and JCL files to the z/OS system, for example, using FTP.
6. On the z/OS system, create datasets to store the object and load modules that are generated by the compilation and link of the program.

   For the C language program:

   a. Execute the CCOMPILE JCL to create the object module.
   b. Execute the CLINK JCL to link the object module and create the load module.
   c. Create the MQ objects required in the z/OS queue manager.
   d. Run the program using the WRCVORD JCL.
   e. Check the output of the program for the messages that show the orders that have been received and processed.
   f. This program runs a loop until a message with the word *shutdown* is received or the operator cancels it.

   For the COBOL language program:

   a. Execute the MQCOBOL JCL to create the object module.
   b. Create the MQ objects required in the z/OS queue manager.
   c. Run the program using the TSTMQCB JCL.
   d. Respond to the prompt message “Enter START, STOP or QUIT” on the operator console with the word “START”. This starts the execution of the callback function. Messages are now retrieved from the queue MATT.WH.ORDERS.
   e. Check the output of the program for the messages that show the orders that have been received and processed.

f. This program runs a loop until the operator responds with the word “QUIT” to the message prompt “Enter START, STOP or QUIT”. The word “STOP” stops the callback function. The callback function can be restarted (START) or terminated (QUIT) by the operator.

19.3 Running the warehousing component

To run the warehousing programs:

1. On the warehouse z/OS system, submit the WRCVORD JCL or TSTMQCB JCL to run the warehouse receive orders program.
2. On the headquarters Windows system, open a command prompt window and execute the **`runas`** command:

   ```
   runas /user:hquser cmd.exe
   ```

3. Go to the command prompt window started by the **`runas`** command.
4. Change the directory to the location of the executable code:

   ```
   cd C:\Scenario\Chapter19_Warehousing\windows
   ```

5. Execute the program Whqsendorders.exe:

   ```
   Whqsendorders MATT.RETAIL.WH.ORDERS QMHQ
   **********************************
   *  Whqsendorders program starts  *
   **********************************
   Connecting to Queue Manager 'QMHQ'
   Opening warehouse retail queue 'MATT.RETAIL.WH.ORDERS' for input
   ```

6. The program output shows the orders that are processed and published to the warehouse:

   ```
   Whqsendorders 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/vegetable/british baby
   carrots,1,101,1215059489750'
   Publishing message:
     Topic string 'matt/warehouse/cat/fresh/vegetable/british baby
   carrots'
     Message data 'cat/fresh/vegetable/british baby
   carrots,1,101,1215059489750'
   ```

19.4 Verifying the warehousing component

To verify correct operation of the warehousing component:

1. On the warehouse z/OS system, submit the JCL to run program WRCVORD or TSTMQCB.
2. On the headquarters Windows system, open a command prompt window and execute the **runas** command:

   ```
   runas /user:hquser cmd.exe
   ```

3. Go to the command prompt window started by the **runas** command. Execute the program Whqsendorders.exe.

   ```
   cd C:\Scenario\Chapter19_Warehousing\windows
   Whqsendorders MATT.RETAIL.WH.ORDERS QMHQ
   ```

4. In another command prompt window, execute the test command file testretailwhordersgen.bat to put some test orders on the MATT.RETAIL.WH.ORDERS queue:

   ```
   cd C:\Scenario\Chapter19_Warehousing\windows
   testretailwhordersgen
   ```

   The output of the command file should be:

   ```
   C:\Scenario\Chapter19_Warehousing\windows>amqsput
   MATT.RETAIL.WH.ORDERS 0<testretailwhordersdata1.txt
   Sample AMQSPUT0 start
   target queue is MATT.RETAIL.WH.ORDERS
   Sample AMQSPUT0 end

   C:\Scenario\Chapter19_Warehousing\windows>pause
   Press any key to continue . . .
   ```

5. Check the output of the Whqsendorders.exe program running in the other window:

   ```
   **********************************
   *  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'
```

6. On the warehouse z/OS system, check the output of the TSTMQCBM COBOL program:

```
TSTMQCBF - *** Warehouse order received ***
 TSTMQCBF - Description: cat/fresh/fruit/apples
 TSTMQCBF - Quantity:    5
 TSTMQCBF - Reference:   ORDER0001
 TSTMQCBF  *** AN ERROR OCCURRED IN MQINQMP. COMPLETION CODE =   2
REASON CODE =  2471 ***
 TSTMQCBF - *** Warehouse order received ***
 TSTMQCBF - Description: cat/tinned/fish/salmon
 TSTMQCBF - Quantity:    3
 TSTMQCBF - Reference:   ORDER0002
 TSTMQCBF  *** AN ERROR OCCURRED IN MQINQMP. COMPLETION CODE =   2
REASON CODE =  2471 ***
 TSTMQCBF - *** Warehouse order received ***
 TSTMQCBF - Description: cat/dry/flour/selfraising
 TSTMQCBF - Quantity:    2
 TSTMQCBF - Reference:   ORDER0003
 TSTMQCBF  *** AN ERROR OCCURRED IN MQINQMP. COMPLETION CODE =   2
REASON CODE =  2471 ***
 TSTMQCBF - *** Warehouse order received ***
 TSTMQCBF - Description: cat/glass/spirit/vodka
 TSTMQCBF - Quantity:    6
 TSTMQCBF - Reference:   ORDER0004
 TSTMQCBF  *** AN ERROR OCCURRED IN MQINQMP. COMPLETION CODE =   2
REASON CODE =  2471 ***
```

Note: The error in MQINQMP (reason code 2471) is caused by the test command file testretailwhordersgen.bat that does not create the message properties that are expected by the COBOL program.

7. On the z/OS system, check the output of the WRCVORD C program:

```
*************************************
 *  Wwhreceiveorders program starts  *
 *************************************
 Main: Connecting to Queue Manager 'MQ71'
 Main: Opening warehouse orders queue 'MATT.WH.ORDERS' for input
 Main: Registering MQ callback function
 Main: Starting message consumer
 Main: While loop sleeping every  10 seconds
 WhOrdersConsumer: Message removed, Reason=0, DataLength=43,
   BufferLength=43, GMO.ReturnedLength=-1
 <*> Warehouse has received an order:
   Description       'cat/fresh/fruit/apples'
   Quantity          '5'
   Reference         'ORDER0001'
 MQIMQMP failed with reason code 2471
 WhOrdersConsumer: Message removed, Reason=0, DataLength=43,
   BufferLength=43, GMO.ReturnedLength=-1
 <*> Warehouse has received an order:
   Description       'cat/tinned/fish/salmon'
   Quantity          '3'
   Reference         'ORDER0002'
 MQIMQMP failed with reason code 2471
 WhOrdersConsumer: Message removed, Reason=0, DataLength=46,
   BufferLength=46, GMO.ReturnedLength=-1
 <*> Warehouse has received an order:
   Description       'cat/dry/flour/selfraising'
   Quantity          '2'
   Reference         'ORDER0003'
 MQIMQMP failed with reason code 2471
 WhOrdersConsumer: Message removed, Reason=0, DataLength=43,
   BufferLength=46, GMO.ReturnedLength=-1
 <*> Warehouse has received an order:
   Description       'cat/glass/spirit/vodka'
   Quantity          '6'
   Reference         'ORDER0004'
 MQIMQMP failed with reason code 2471
```

Note: The error in MQINQMP (reason code 2471) is caused by the test command file testretailwhordersgen.bat that does not create the message properties that are expected by the C program.

19.5 Summary

The warehousing component uses a durable non-managed subscription that is created using administration commands 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.

Callback for asynchronous consume is demonstrated on a z/OS environment in both C and COBOL languages. A benefit of callback is that message buffers are allocated by the queue manager with the correct size for the message length. It also allows the main line of the program to perform other useful work, rather than waiting for messages to arrive on a call to MQGET.

The use of new MQI functions is also demonstrated to inquire about message properties.

A

Scenario preparation scripts

This appendix contains scripts that are used for common scenario preparation:

- QMHQobjects.txt: The MQSC command script for headquarters queue manager object definitions
- QMWHobjects.txt: The MQSC command script for warehouse queue manager object definitions
- QMHQsetaut.bat: The Windows batch script for headquarters queue manager object authority settings, using the **`setmqaut`** command

For detailed information about these scripts refer to 14.2, “WebSphere MQ objects setup” on page 278.

QMHQobjects.txt

Example A-1 contains the QMHQobjects.txt script.

Example: A-1 The script QMHQobjects.txt

```
* This MQSC script defines headquarters queue manager objects
* needed for the scenario
* The script uses REPLACE so it can be run repeatedly
* V1, Martin Cernicky, 12 Dec 2007

* 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) +
   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

Example A-2 contains the QMWHobjects.txt script.

Example: A-2 The script QMWHobjects.txt

```
* This MQSC script defines warehouse queue manager objects
* needed for the scenario
* The script uses REPLACE so it can be run repeatedly
* V1, Martin Cernicky, 12 Dec 2007
* V2, Glenn Baddeley, 16 Jul 2008

* 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(QMHQ) +
   USAGE(XMITQ) +
   REPLACE
DEFINE QLOCAL(MATT.WH.ORDERS) +
   REPLACE

* Channels
DEFINE CHANNEL(QMWH_TO_QMHQ) +
   CHLTYPE(SDR) +
   TRPTYPE(TCP) +
   CONNAME('sam725hq(1414)') +
   XMITQ(QMHQ) +
   REPLACE
DEFINE CHANNEL(QMHQ_TO_QMWH) +
   CHLTYPE(RCVR) +
   TRPTYPE(TCP) +
   REPLACE

* End of script
```

QMHQsetaut.bat

Example A-3 contains the QMHQsetaut.bat script.

Example: A-3 The script QMHQsetaut.bat

```
@echo off
rem  This script sets up headquarters queue manager authorities
rem  needed for the scenario
rem  The script removes all authorities at first so it
rem  can be run repeatedly
rem  V1, Martin Cernicky, 12 Dec 2007

rem   Set up for group "hq"
setmqaut -m QMHQ -t qmgr -g hq +none
setmqaut -m QMHQ -t qmgr -g hq +connect +inq
setmqaut -m QMHQ -t queue -n MATT.RETAIL.ORDERS -g hq -remove
setmqaut -m QMHQ -t queue -n MATT.RETAIL.ORDERS -g hq +get +inq +browse
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.SUPPLIER.QUOTES -g hq -remove
setmqaut -m QMHQ -t queue -n MATT.SUPPLIER.QUOTES -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
+resume
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 +inq
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 +inq
+browse
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 IBM Redbooks publication 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
Chapter14_ScenarioPrep.zip	Chapter 14 zipped code samples
Chapter15_SupplierPricing.zip	Chapter 15 zipped code samples
Chapter16_StoreOrdering.zip	Chapter 16 zipped code samples
Chapter17_News.zip	Chapter 17 zipped code samples
Chapter18_WebOrdering.zip	Chapter 18 zipped code samples
Chapter19_Warehousing.zip	Chapter 19 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.

Asynchronous_Actions that occur without any constraints on timing, for example, sending a message to an independent party which may or may not be active. Compare with Synchronous.

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.

Browse_In terms of WebSphere MQ, to examine the contents of a message without destructively removing it from the queue.

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.

Callback_A software technique which allows a function to be called when a defined event occurs, independent of other threads of the program.

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.

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.

Co-existence_In terms of WebSphere MQ for z/OS, the ability to install and run more than one version of the product in one operating system image. Co-existence does not exist on any other platforms supported by WebSphere MQ.

Common Object Request Broker Architecture (CORBA)_A standard for software calls that enables programs to communicate across multiple languages and computers.

Conversation_ In terms of network communication, a conversation is an established conduit for the exchange of information between two computer programs.

Customer Information Control System (CICS)_ IBM proprietary transaction server that runs on z/OS systems.

Data conversion_The process of converting the binary representation of characters and numbers from that of one environment to that of another.

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.

Distributed queuing_Using particular types of WebSphere MQ channels to interconnect queue managers, namely SENDER, RECEIVER, SERVER and REQUESTOR. Compare with queue manager cluster.

Eclipse_An open-source extensible software platform for integrated development and management of applications primarily written in Java, led by the Eclipse Foundation.

Engine_An internal software component which performs common functions which are used throughout a product, for example Publish/Subscribe in WebSphere MQ.

Enterprise Service Bus (ESB)_A software architecture where messaging integration technologies are used, generally as part of a SOA implementation.

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.

Full duplex_A communication system involving two connected parties where information can be sent in either direction at any time. Compare with Half duplex

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.

Half duplex_A communication system involving two connected parties where information can only be sent in one direction at a time. Compare with Full duplex.

Handle_A binary number that represents an active context or reference to an entity, such as a connection, an open queue, or a message in WebSphere MQ.

Heartbeat_In WebSphere MQ, a flow which is periodically sent over channels when there are no MQ messages to send. This allows channel failures to be detected earlier.

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.

Hyper Text Transport Protocol (HTTP)_A protocol for client-server communication which has been popularized by Internet for Web servers to deliver content to Web browser clients.

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 standardized application programming interface for the Java programming language, which is part of the Java 2 platform Enterprise Edition standard.

Message_A piece of information, with addressing or other meta information associated, that can be passed between software components.

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 property_A name and associated value that is stored a WebSphere MQ message, separated from the application message data.

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.

Message selector_A logical combination of message property names and values that defines criteria for selecting messages from a queue.

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.

Non-persistent message_A WebSphere MQ message which is not recovered if a queue manager fails, or is restarted, or if there is a failure in the underlying infrastructure or operating system. Compare with Persistent message.

Object Authority Manager (OAM)_A component of a WebSphere MQ queue manager that performs authority checking.

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.

Persistent message_A WebSphere MQ message which can be fully recovered if a queue manager fails, or is restarted, or if there is a failure in the underlying infrastructure or operating system. Persistent messages are required for guaranteed message delivery in WebSphere MQ. Compare with Non-persistent message.

Personal certificate_A public certificate which can be used to identify an entity, combined with the private key for that certificate.

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

Process_An executing instance of a computer program, consisting of one or more threads and various resources, managed by an operating system.

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.

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 using CLUSSDR and CLUSRCVR channels, which simplifies administration and provides workload balancing facilities for scalability and service availability. Compare with Distributed queuing.

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.

Remote Procedure Call (RPC)_Calling a software component from a program as though it were local, but the called procedure may be executed on another computer. The calling program waits for the called procedure to complete execution before continuing.

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 Access Control Facility (RACF)_IBM proprietary security management service that runs on z/OS systems

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 performance.

Secure Sockets Layer (SSL)_An industry standardized technology to provide authentication and secure 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-oriented architecture (SOA)_A software architecture where business processes are grouped and interoperate as loosely coupled services. SOA has well-defined guiding principles and architectural constructs.

Servlet_A Java object in a Web server that receives requests and generates replies, typically involving HTML content.

Socket_A logical end-point in a connection between two processes over the TCP/IP network communications protocol.

State information_Information that changes over time, but only has one value at any point in time.

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.

Synchronous_Actions that occur with constraints on timing, for example, invoking a Remote Procedure Call where the other program must be available to immediately service the call. Compare with Asynchronous.

Thread_A single independent sequence of executing instructions in a Process.

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 standardized technology to provide authentication and secure communication.

Unit of work_A logical grouping of actions, which must either all succeed or all fail.

Web services_A standardized way to describe and invoke services.

WebSphere MQ Client_The software components of WebSphere MQ which allow client applications to connect to a local or remote queue manager over client connection channels.

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).

WebSphere MQ Server_The software components of WebSphere MQ which allow a queue manager instance to be run on the local machine. Server also includes support for local Clients.

Abbreviations and acronyms

API	Application programming interface
CAF	Client Attach Facility
CCDT	Client Channel Definition Table
CCSID	coded character set identifier
CHIN	channel initiator
CL	Command Language on i5/OS
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CSV	Comma-separated values
DLL	Dynamic Link Library
DoS	Denial of Service
DST	Daylight Saving Time
ESB	Enterprise Service Bus
FTP	File Transfer Protocol
GTS	Global Technology Services
GUI	Graphical User Interface
HTTP	Hyper Text Transport Protocol
IBM	International Business Machines Corporation
ISV	Independent Software Vendor
IT	information technology
ITSO	International Technical Support Organization
JAR	Java Archive
JCA	Java Connector Architecture
JCL	Job Control Language
JMS	Java Message Service
MCA	Message Channel Agent
MDB	Message Driven Beans
MLP	Message Listener Port
MOM	Message-oriented middleware
MQI	Message Queue Interface
MQMD	MQ Message Descriptor
OAM	Object Authority Manager
OTE	Open Transaction Environment
PCF	Programmable Command Format
PTF	Program Temporary Fix
QoS	Quality of service
QR	Quasi-Reentrant
QSG	Queue Sharing Group
REST	Representational State Transfer
RFH	Rules and Formatting Header
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
SYSOUT	Standard output of the program on z/OS
TCB	Task Control Block
TCP/IP	Transmission Control Protocol / Internet Protocol
TLS	Transport Layer Security
UK	United Kingdom
UOW	Unit Of Work
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

WAS	IBM WebSphere Application Server
XML	Extensible Markup Language
XMS	IBM message service client

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this IBM Redbooks publication.

IBM Redbooks publications

For information about ordering these publications, see "Sun Java" on page 391. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere MQ V6 Fundamentals*, SG24-7128

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

Other publications

These publications are also relevant as further information sources from the WebSphere MQ V7.0 Information Center:

- *WebSphere MQ Publish/Subscribe User's Guide V7.0*
- *WebSphere MQ Messages V7.0*
- *WebSphere MQ Intercommunication V7.0*
- *WebSphere MQ Clients V7.0*
- *WebSphere MQ Script (MQSC) Command Reference V7.0*
- *WebSphere MQ System Administration Guide V7.0*
- *WebSphere MQ i5/OS System Administration Guide V7.0*
- *WebSphere MQ Programmable Command Formats and Administration Interface V7.0*
- *WebSphere MQ System Administration Guide V7.0*
- *WebSphere MQ z/OS Concepts and Planning Guide V7.0*
- *WebSphere MQ z/OS Systems Setup Guide V7.0*
- *WebSphere MQ z/OS Program Directory V7.0*
- *WebSphere MQ Publish/Subscribe Users Guide V7.0*

- *WebSphere MQ Security V7.0*
- *WebSphere MQ Application Programming Guide V7.0*
- *WebSphere MQ Application Programming Reference V7.0*
- *WebSphere MQ Migration Information V7.0*
- *WebSphere MQ Quick BeginningsV7.0 (for each distributed platform)*
- *WebSphere MQ Using Java V7.0*
- *WebSphere MQ Using C++ V7.0*
- *WebSphere MQ Using .NET V7.0*
- *WebSphere MQ Web Services V7.0*

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp

These publications are also available as PDF files from the WebSphere MQ V7.0 library:

http://www.ibm.com/software/integration/wmq/library

Online resources

These Web sites are also relevant as further information sources:

- Information on SOA

  http://www.ibm.com/soa
- WebSphere MQ product information

  http://www.ibm.com/support/docview.wss?uid=swg27007431
- WebSphere MQ system requirements:

  http://www.ibm.com/software/integration/wmq/requirements
- WebSphere MQ File Transfer Edition V7.0

  http://www.ibm.com/software/integration/wmq/filetransfer
- WebSphere MQ Bridge for HTTP

  http://www.ibm.com/software/integration/wmq/httpbridge/
- WebSphere Application Server Installation Guide

  http://www.ibm.com/software/webservers/appserv/was/library/
- SupportPac MS0Q: WebSphere MQ Explorer Publish/Subscribe plug-in

  http://www.ibm.com/support/docview.wss?uid=swg24013508

- SupportPac MS0B: WebSphere MQ Java classes for PCF

  `http://www.ibm.com/support/docview.wss?uid=swg24000668`
- SupportPac MA0Y: IBM WebSphere MQ Bridge for HTTP

  `http://www.ibm.com/support/docview.wss?uid=swg24016142`
- SupportPac MO72: MQSC Client for WebSphere MQ

  `http://www.ibm.com/support/docview.wss?uid=swg24007769`
- SupportPac MQC6: WebSphere MQ V6.0 Clients

  `http://www.ibm.com/support/docview.wss?uid=swg24009961`
- WebSphere MQ V7.0 Application Programming Guide → Compilation of a C program

  `http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp`
- Sun Java

  `http://www.java.com`
- JMS Specification

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

These pages in the WebSphere MQ V7.0 Information Center are used:

- WebSphere MQ V7.0 product installation and quick beginnings guides
- WebSphere MQ V7.0 Using Java → Mapping JMS messages
- WebSphere MQ V7.0 Using Java → WebSphere MQ classes for JMS → Properties of objects
- WebSphere MQ V7.0 Using Java → WebSphere MQ classes for JMS → Using WebSphere MQ classes for JMS → Solving problems
- WebSphere MQ V7.0 Intercommunication → Implications of sharing conversations
- WebSphere MQ V7.0 Clients → Using MQCONN calls
- WebSphere MQ V7.0 System Administration → Details on the MaxChannels, MaxActiveChannels, MAXCHL and MAXTCP parameters
- WebSphere MQ V7.0 Web Services → WebSphere MQ Bridge for HTTP → HTTP Return codes

How to get Redbooks publications

You can search for, view, or download Redbooks publications, Redpapers publications, 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

WebSphere MQ V7.0 Features and Enhancements

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

This IBM® Redbooks® publication is divided into three parts:

- Part 1, "Introduction" on page 1, provides an introduction to message-oriented middleware and the WebSphere® MQ product. We discuss the concept of messaging, 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, "WebSphere MQ V7.0 enhancements and changes" on page 41, explains the new WebSphere MQ V7.0 features and enhancements in detail and includes compatibility and the migration considerations from the previous supported versions.
- Part 3, "Scenario" on page 253, contains a scenario that 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 379.

SG24-7583-00 ISBN 0738432202