[MS-MQMA]:
Message Queuing (MSMQ):
Architecture Protocol Specification

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
05/11/2007	0.1		MCPP Milestone 4 Initial Availability
08/10/2007	1.0	Major	Updated and revised the technical content.
09/28/2007	1.0.1	Editorial	Revised and edited the technical content.
10/23/2007	1.0.2	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
11/30/2007	1.0.3	Editorial	Revised and edited the technical content.
01/25/2008	1.0.4	Editorial	Revised and edited the technical content.

Table of Contents

1 Introduction 5
1.1 Glossary 5
1.2 References 6
1.2.1 Normative References 6
1.2.2 Informative References 6
1.3 Relationship to Other Protocols 7
1.4 Applicability Statement 8

2 Product Architecture 9
2.1 Introduction 9
2.2 Application Profiles 9
2.2.1 Independent Client Profile 9
2.2.1.1 Key Roles and Concepts 10
2.2.1.1.1 Independent Client 10
2.2.1.1.2 Workgroup Mode 10
2.2.2 Dependent Client Profile 11
2.2.2.1 Key Roles and Concepts 12
2.2.2.1.1 Dependent Client 12
2.2.2.1.2 Supporting Server 12
2.2.2.1.3 MSMQ Directory Service 12
2.2.2.1.4 Domain Mode 12
2.2.3 Dependent Client Over DCOM Profile 12
2.2.4 Management Client Query Profile 13
2.2.4.1 Key Roles and Concepts 14
2.2.4.1.1 Management Client 14
2.2.5 Management Client Update Profile 14
2.2.6 Transaction and Routing Profile 15
2.2.6.1 Key Roles and Concepts 17
2.2.6.1.1 Routing Server 18
2.2.6.1.2 Distributed Transaction Coordinator 18
2.3 MSMQ Directory Service 18

3 Protocol Architecture 19
3.1 Protocol Overview 19
3.1.1 Message Queuing Binary Protocol 19
3.1.1.1 Overview 19
3.1.1.2 Relationship to Other Protocols 19
3.1.2 Directory Service Protocol 19
3.1.2.1 Overview 19
3.1.2.2 Relationship to Other Protocols 20
3.1.3 Directory Service Discovery Protocol 20
3.1.3.1 Overview 20
3.1.3.2 Relationship to Other Protocols 20
3.1.4 Queue Manager to Queue Manager Protocol 20
3.1.4.1 Overview 20
3.1.4.2 Relationship to Other Protocols 20
3.1.5 Queue Manager Client Protocol 21
3.1.5.1 Overview 21
3.1.5.2 Relationship to Other Protocols 21
3.1.6 Queue Manager Management Protocol 21
3.1.6.1 Overview 21
3.1.6.2 Relationship to Other Protocols 21

3.1.7 Directory Service Change Notification Protocol 21
3.1.7.1 Overview 21
3.1.7.2 Relationship to Other Protocols 22
3.1.8 Queue Manager Remote Read Protocol 22
3.1.8.1 Overview 22
3.1.8.2 Relationship to Other Protocols 22
3.2 Protocol Usage 22
3.2.1 MSMQ Clients 22
3.2.2 MSMQ Server 22
3.2.3 MSMQ Version 1.0 and MSMQ Version 2.0 Protocol Architecture 23
3.2.4 MSMQ Version 3.0 and Version 4.0 Protocol Architecture 24
4 Index 27

1 Introduction

The main objective of this document is to provide the reader with a better grasp of how the **Message Queuing (MSMQ)** protocols fit together.

To better understand the MSMQ protocols, this document will start with a brief overview of the MSMQ application profiles, followed by an introduction to the various protocols that are involved in facilitating the client and server interaction of an MSMQ product as well as the overall application architecture.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Access Control List (ACL)
Active Directory (AD)
Globally Unique Identifier (GUID)

The following terms are defined in [MS-MQMQ]:

Cursor
Dependent Client
Direct Format Name
Distribution List
Independent Client
Message
MSMQ Client
MSMQ Directory Service (MQDS)
MSMQ Queue Manager
MSMQ Supporting Server
Path Name
Private Queue
Primary Enterprise Controller (PEC)
Primary Site Controller (PSC)
Public Queue
Queue
Queue Alias
Transactional Queue

The following terms are specific to this document:

Asynchronous Messaging: Communication between two applications or systems, independent of time.

Backup Site Controller (BSC): A server that retains a copy of the **Primary Site Controller (PSC)** belonging to its **site** and functions as a backup in case the **PSC** fails.

Directory Service: A distributed data storage system that allows computers joined to its network to store, edit, and retrieve information.

Domain Mode: The operation of **message** queuing clients or servers integrated with **MSMQ Directory Service (MQDS)**.

Down-Level Service: An **MSMQ** component that provides **MQDS** support for **MSMQ** version 1.0 or **MSMQ** version 2.0 clients. This service acts as a local relay between **Active Directory**

and the Message Queuing (MSMQ): Directory Service Protocol. The Down-Level service must be collocated on an **Active Directory** domain controller.

Guaranteed Message Delivery: **Messages** are stored on a disk-based **queue** by a **queue manager (QM)** and then forwarded to the destination by the **QM** when network connectivity is restored. This provides a guarantee that the **messages** will be delivered to the receiver as soon as the **QM** has connectivity.

Management Client: An **MSMQ client** that performs management operations, as specified in [MS-MQMR].

Microsoft Message Queuing (MSMQ): A communications service that provides asynchronous and reliable **message** passing between distributed client applications. In **message** queuing, clients send **messages** to **queues** and consume **messages** from **queues**. The **queues** provide persistence of the **messages**, enabling the sending and receiving client applications to operate asynchronously from each other.

Outgoing Queue: A **queue** on a **QM** where **messages** are held until they can be transferred to the next **queue manager**.

Queue Manager (QM): See **MSMQ Queue Manager**.

Queue Name: A general term used to describe the various ways to designate **MSMQ queues**. **Queues** can be specified using **path names**, format names, and **queue aliases**. For more information, see [MS-MQMQ] section 2.1.

Site: See **MSMQ Site**.

Site Link: See **MSMQ Site Link**.

Supporting Server: See **MSMQ Supporting Server**.

Transaction: An atomic **transaction** context dispensed by a **transaction coordinator**, such as Microsoft Distributed Transaction Coordinator (for more information, see [MSDN-DTC]), and used by a **queue manager** to coordinate its state changes with state changes in other resource managers. For more information, see [MS-DTCO].

Transaction Coordinator: Provides concrete mechanisms for beginning, propagating, and completing atomic **transactions**. It also provides mechanisms for coordinating agreement on a single atomic outcome for each **transaction**, and for reliably distributing that outcome to all participants in the **transactions**. For more information, see [MS-DTCO].

Workgroup mode: The operation of **message queuing** without **MSMQ Directory Service (MQDS)** integration.

1.2 References

1.2.1 Normative References

None.

1.2.2 Informative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site,

http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[LDAP] Microsoft Corporation, "About Lightweight Directory Access Protocol", http://msdn2.microsoft.com/en-us/library/aa366075.aspx

If you have any trouble finding [LDAP], please check here.

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification", June 2007.

[MS-DTCO] Microsoft Corporation, "MSDTC Connection Manager: OleTx Transaction Protocol Specification", July 2007.

[MS-MQBR] Microsoft Corporation, "Message Queuing (MSMQ): Binary Reliable Messaging Algorithm", September 2007.

[MS-MQCN] Microsoft Corporation, "Message Queuing (MSMQ): Directory Service Change Notification Protocol Specification", September 2007.

[MS-MQDS] Microsoft Corporation, "Message Queuing (MSMQ): Directory Service Protocol Specification", July 2007.

[MS-MQMP] Microsoft Corporation, "Message Queuing (MSMQ): Queue Manager Client Protocol Specification", August 2007.

[MS-MQMQ] Microsoft Corporation, "Message Queuing (MSMQ): Data Structures", August 2007.

[MS-MQMR] Microsoft Corporation, "Message Queuing (MSMQ): Queue Manager Management Protocol Specification", August 2007.

[MS-MQQB] Microsoft Corporation, "Message Queuing (MSMQ): Message Queuing Binary Protocol Specification", August 2007.

[MS-MQQP] Microsoft Corporation, "Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol Specification", August 2007.

[MS-MQRR] Microsoft Corporation, "Message Queuing (MSMQ): Queue Manager Remote Read Protocol Specification", June 2007.

[MS-MQSD] Microsoft Corporation, "Message Queuing (MSMQ): Directory Service Discovery Protocol Specification", August 2007.

[MSDN-DTC] Microsoft Corporation, "Distributed Transaction Coordinator", http://msdn2.microsoft.com/en-us/library/ms684146.aspx

[PCT1] Benalogh, J., Lampson, B., Simon, D., Spies, T., and Yee, B., "The Private Communication Technology (PCT) Protocol", October 1995, http://tools.ietf.org/html/draft-benaloh-pct-00

If you have any trouble finding [PCT1], please check here.

1.3 Relationship to Other Protocols

This document does not depend on any other protocol. It describes how the MSMQ family of protocols is used together in sample scenarios.

1.4 Applicability Statement

This document is not a protocol specification; its main objective is to provide an architectural overview of the MSMQ protocols. This document is informative in nature.

This document covers the versions of Microsoft Message Queuing (MSMQ) that were shipped in the following versions of Windows.

MSMQ version	Windows version
MSMQ version 4.0	Windows Server 2008, Windows Vista
MSMQ version 3.0	Windows Server 2003, Windows XP
MSMQ version 2.0	Windows 2000
MSMQ version 1.0	Windows NT 4.0

2 Product Architecture

2.1 Introduction

Message Queuing (MSMQ) technology enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. These applications send **messages** to **queues** and read messages from queues. MSMQ provides the message queuing services for creating distributed and **asynchronous messaging** systems. Some of its features include **guaranteed message delivery**, routing, and security.

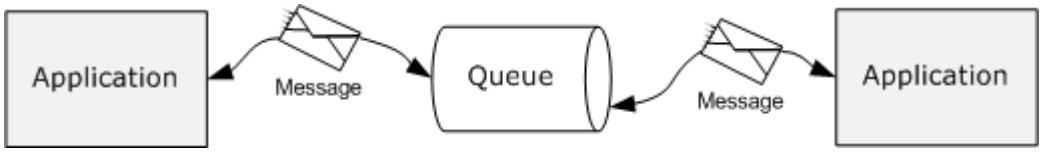

Figure 1: Using message queuing with applications

2.2 Application Profiles

This section describes several application profiles that depend on MSMQ to send queued messages. An application profile consists of two or more MSMQ systems representing MSMQ roles. These application profiles show the interaction between each of the roles, as well as the specific protocol that facilitates that interaction. The following sections present several profiles, introduce key concepts, and highlight one possible action in that configuration. The entire range of communication possible between two roles in a profile is described in the corresponding protocol specification.

The profiles start with the most basic and increase in complexity to profiles that require almost all MSMQ protocols in order to provide a set of functionalities. The main purpose is to review specific MSMQ functionalities and to show which protocols are needed to enable them, and thus to help explain how the underlying protocols fit together.

2.2.1 Independent Client Profile

The most basic queuing profile consists of two applications that can send messages to and receive messages from each other, as shown in the following diagram. The message transfer is an asynchronous process, where the Initiating Application sends a message to its **queue manager (QM)** and continues. The send operation terminates when the sending QM accepts the message; the sending QM then holds the message in an **outgoing queue** and sends it to its destination when the QM establishes a direct connection. Similarly on the receiving side, the receiving QM holds the message in a queue until the Receiving Application reads the message.

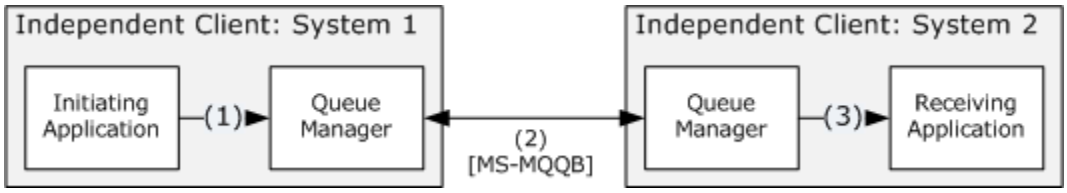

Figure 2: Asynchronous messaging with queue managers

In this case the Initiating Application on System 1 needs to be aware of the **direct format name** of the queue on the QM on System 2. Both of the systems are referred to as **independent clients**. Furthermore, in this scenario they do not need to interact with an **MSMQ Directory Service**

(MQDS) in order to send and receive messages. If these systems do not have a **directory service** integration installed, they are said to be operating in the **Workgroup Mode**. Following are the set of actions that the initiating application takes in sending a queued message to the receiving application.

The Initiating Application on System 1 creates a message and sends it to the local QM using the MSMQ application programming interfaces (APIs). The application provides the QM with the required information such as the direct format name, message properties, and the message body etc.

The QM in System 1 uses the information received from the Initiating Application to package the message and sends it to the destination queue located on the receiving application's machine. This transfer happens as soon as the QM on System 1 has connectivity with the QM on System 2; until then the messages are stored in an outgoing queue. For more information about this protocol transfer, see [MS-MQQB].

The Receiving Application uses the MSMQ runtime APIs to access the queue on the System 2's QM and receives the message.

2.2.1.1 Key Roles and Concepts

The profile describes the MSMQ Independent Client role as well as the Workgroup Mode operation concept, which are explained in the following sections.

2.2.1.1.1 Independent Client

An independent client is any MSMQ installation with a QM collocated on the same host machine. Computers configured with independent client functionality run the MSMQ runtime, host queues, send and receive messages, and can operate while disconnected from the network. Independent clients do not require synchronous access to other Message Queuing servers to perform these actions.

If an independent client is disconnected from the network, it can continue to generate messages destined for other QMs. The independent client stores these messages locally and sends them as soon as the network connection is re-established. Similarly a QM that is receiving messages holds them in a queue until an application is ready to receive them.

2.2.1.1.2 Workgroup Mode

MSMQ independent clients operating without Directory Service integration are said to be operating in workgroup mode. While operating in this mode, MSMQ's functionality is restricted; it cannot take advantage of features provided by a Message Queuing Down Level (MQDS), such as routing, looking up **public queues**, and some security functionalities.

An MSMQ service running in workgroup mode is also restricted to creating and managing **private queues** on the local system and sending and receiving messages using direct format names. Independent clients in workgroup mode can send messages to other queues in the same workgroup, to queues in other workgroups, and generally to any queue specified by a direct format names. Note that public queues, registered in an MQDS, can be accessed directly via a direct format name also. Since the direct format name contains the physical address of the queue, a client operating in workgroup mode is able to access the public queue without querying the MQDS. Workgroup mode functionality is available in MSMQ versions 2.0, 3.0, and 4.0.

2.2.2 Dependent Client Profile

A **dependent client** consists of an application hosted on a dependent client sending messages to an application hosted on a separate machine. It introduces three additional MSMQ roles:

- A dependent client
- A **supporting server**
- An MQDS

The difference between this profile and the previous one is that in this profile, the dependent client uses a supporting server to send and receive messages and queries the MQDS for information required to send these messages.

The initiating application needs to know the **queue name** of the queue from which the receiving application receives messages. In this profile the initiating application only knows the **path name** of the public queue. The QM on the Supporting Server can locate the queue based on its path name by querying the MSMQ directory service. MSMQ is said to be operating in **domain mode** when integrated with an MQDS. In MSMQ version 1.0, Dependent Clients can locate supporting servers, and QMs can locate the MQDS Servers via Message Queuing (MSMQ): Directory Service Discovery Protocol; however, in MSMQ version 2.0 onwards, the supporting server needs to be specified during installation. Furthermore, in MSMQ version 2 onwards, MSMQ uses **Active Directory** to implement the MQDS, and QMs locate Active Directory as specified in [MS-ADTS] section 7.3.

In this scenario, the Dependent Client on System 1 is configured to use the Supporting Server on System 2. The QMs on the Supporting Server and Independent client are aware of the MQDS. Furthermore, the QM on the Independent Client is hosting a public queue and it is registered in the MQDS. Following are the steps needed by the initiating application to send messages to the receiving application.

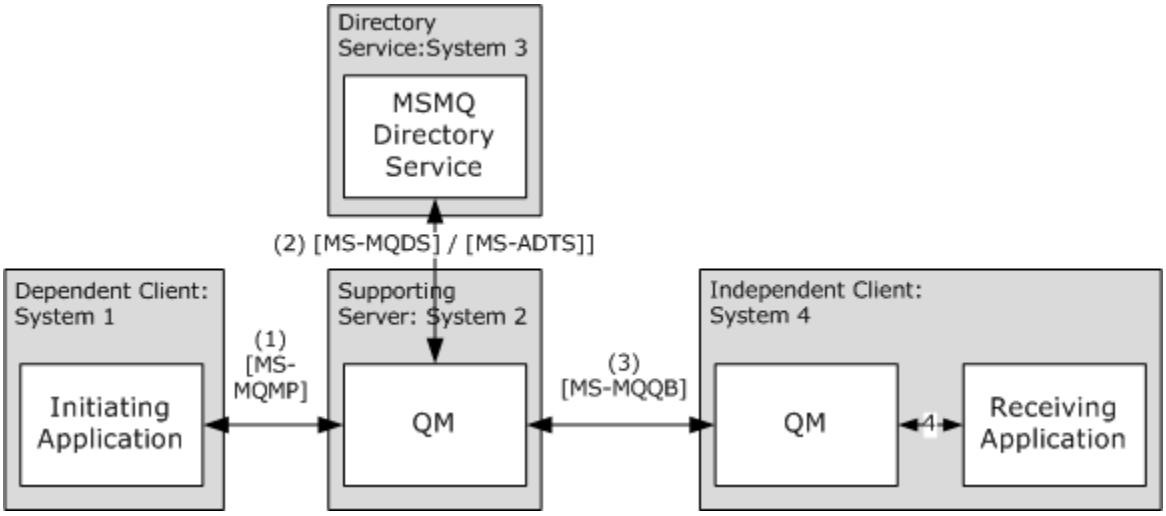

Figure 3: Dependent client using supporting server

1. The Initiating Application on the Dependent Client uses the MSMQ runtime to access the QM in the Supporting Server. The communication between the Dependent Client and Supporting server happens over Message Queuing (MSMQ): Queue Manager Client Protocol. In this case, the initiating application sends the path name of the public queue hosted on System 4 as well as the message to the supporting server to process.

2. The QM on the Supporting Server obtains information about the public queue on System 4 from the MSMQ Directory Service. This exchange is specified in [MS-MQDS] for MSMQ version 1.0 or in [MS-ADTS] for MSMQ version 2.0 onwards.

3. The QM on the Supporting Server sends the message to the QM on the Independent Client on System 4 over the Message Queuing (MSMQ): Message Queuing Binary Protocol.

4. The Receiving Application accesses the message through MSMQ API calls to the QM.

2.2.2.1 Key Roles and Concepts

This application profile details three key roles for **MSMQ**: Dependent Clients, Supporting Server and MQDS. It also describes the MSMQ domain mode operation. These roles and concepts are further explained in the following sections.

2.2.2.1.1 Dependent Client

An **MSMQ client**, referred to as a dependent client, is any system that has the MSMQ runtime installed. The system has access to queuing functionality through the MSMQ APIs. The distinguishing feature of the dependent client is that there is no queue manager installed on the system. The system therefore requires a direct connection to an **MSMQ supporting server** to send or receive MSMQ messages.

2.2.2.1.2 Supporting Server

A supporting server is an MSMQ installation that provides MSMQ queuing functionality for the dependent clients it supports. It hosts queues and stores, receives, and sends messages for its dependent clients. A supporting server can only be installed on a server operating system.

2.2.2.1.3 MSMQ Directory Service

The MSMQ directory service is a network directory service that stores information pertinent to MSMQ in an enterprise. MSMQ relies on information stored in the MQDS to route messages through routing servers, authenticate and encrypt messages, access public queues, and **queue aliases**, **distribution lists**, and **Access Control Lists (ACLs)**. Information stored in the AD is described in the following section.

2.2.2.1.4 Domain Mode

Domain Mode operation refers to the integration of an MQDS with MSMQ. An MSMQ operating in domain mode allows machines to store and query for information in a networked directory service. Certain MSMQ features are only available when a machine is integrated with an MQDS, such as routing and public queues.

2.2.3 Dependent Client Over DCOM Profile

The Dependent Client profile described in section 2.2.2 can also be executed using MSMQ's ActiveX Protocol interface described in [MC-MQAC]. This interface consists of several Distributed Component object model (DCOM) [MS-DCOM] components that can be leveraged by an application to perform MSMQ functionality. For example, a client application can use [MC-MQAC] to send and receive messages from a remote machine that has MSMQ with a queue manager installed. The [MC-MQAC] protocol lists the full set of the operations exposed.

The key difference between this profile and the Dependent Client Profile in section 2.2.2 is that there is no designated supporting server required. Any independent client that supports the [MC-MQAC]

interface can be used to support a remote client utilizing this interface. The independent client supporting the remote application can be either in domain mode or in workgroup mode.

In this scenario (shown in the following figure), the Independent Client on System 3 is sending messages to the Independent Client on System 2. All of these machines are assumed to be operating in workgroup mode. The Application on System 1 must provide the identifying information for the Queue on Machine 2; similarly the Independent Client on System 3 must know indentifying queue information on System 2 to send messages.

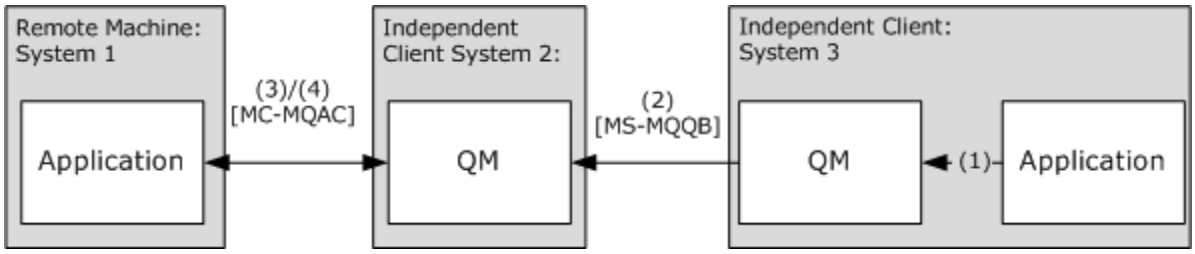

Figure 4: Remote MSMQ operations over ActiveX

1. The Initiating Application on System 3 creates a message and sends it to the local QM using the MSMQ APIs. The application provides the QM with the required information such as the direct format name, message properties, and the message body etc.

2. The QM on System 3 sends a message to the Queue located on the QM in System over [MS-MQQB].

3. The Application on System 1 calls the receive message functionality supported by the ActiveX protocol over [MC-MQAC] on the QM on System 2.

4. The QM on System 2 responds appropriately to the requested action.

2.2.4 Management Client Query Profile

This profile describes how a **Management Client** can query other QMs for information. It consists of a Management Client discovering available QMs and then querying a particular QMs for information. In this profile a Management Client obtains the path names of all private queues from a given QM. In this profile it is assumed that the Management Client and the Independent Client are aware of the MQDS and are operating in the Domain Mode. The details on how an MSMQ machine can locate MQDS are specified in [MS-MQSD] for MSMQ v1.0 or through Lightweight Directory Access Protocol as specified in [MS-ADTS]. Following are the set of actions that facilitate the management client's query.

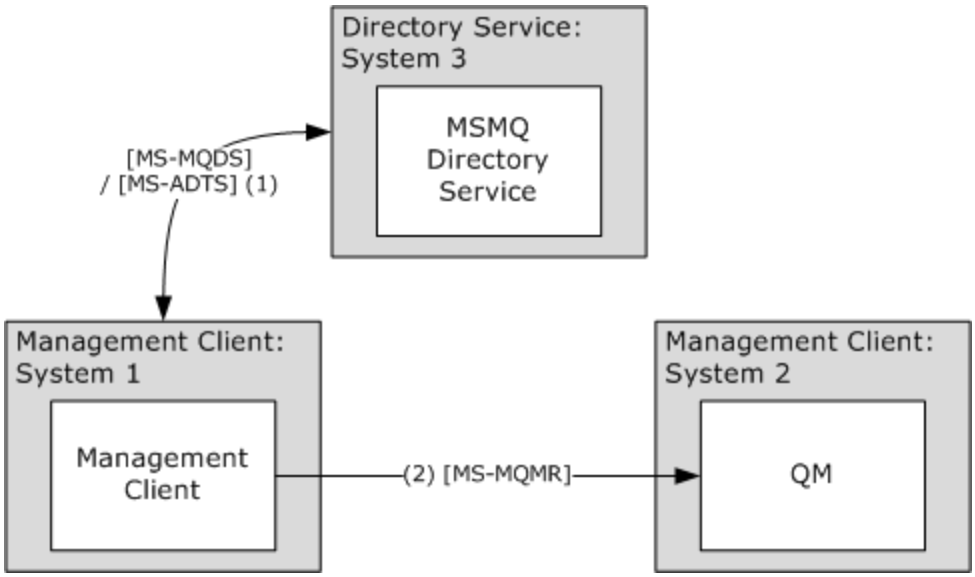

Figure 5: Management client queries

1. The Management Client obtains information about computers with MSMQ functionality by accessing the directory service, using [MS-ADTS] or [MS-MQDS]. The protocol used depends on the version of MSMQ, with [MS-MQDS] used for MSMQ version 1.0 or Lightweight Directory Access Protocol, as specified in [MS-ADTS] for MSMQ version 2.0 onwards. [MS-MQDS] and [MS-ADTS] can also be used to perform other operations on respective directory service objects. Please refer to the protocol documentation for more information.

2. The Management Client requests the QM on System 2 for a list of all private queues hosted by the QM on System 2 using [MS-MQMR]; the QM on System 2 responds over the same protocol.

2.2.4.1 Key Roles and Concepts

The Management Client role of MSMQ is the additional role covered in this profile. The Message Queuing (MSMQ): Queue Manager Management Protocol, as specified in [MS-MQMR], can be used to perform other actions on a remote QM; for more information refer to the protocol documentation.

2.2.4.1.1 Management Client

The management client is an MSMQ client role that performs management operations. The management client functionality is specified in [MS-MQMR].

2.2.5 Management Client Update Profile

This application profile describes how an application can create a public queue on a remote QM. The application profile consists of an MQDS and two independent clients; these clients are aware of the MQDS and can access it. The following diagram and steps describe how an application can create a public queue on a remote machine.

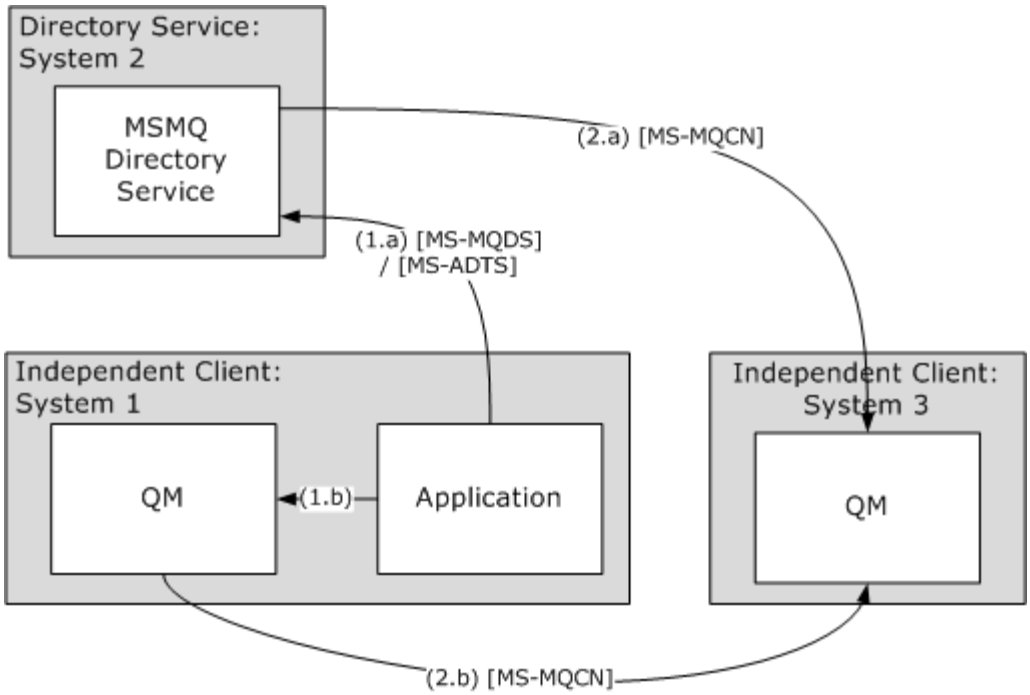

Figure 6: Remote creation of a public queue

1. An application on System 1 requests the creation of a public queue on System 3. For this update to happen, a queue object needs to be created in the MQDS, and the QM on system 3 needs to be notified.

    1. The Application directly updates the MQDS through the MSMQ runtime. This communication happens over [MS-MQDS] for MSMQ version 1.0, and over [MS-ADTS] for MSMQ version 2.0 onwards. This action is shown in the preceding diagram as 1.a.

    2. The application then sends a request to create a public queue to its local QM. This action needs to happen only if the directory service is not MSMQ version 1.0. This action is shown in the preceding diagram as 1.b.

2. The actual creation of the public queue is handled via the QM on System 3. The QM is notified via the Message Queuing (MSMQ): Directory Service Change Notification Protocol; however, the initiation of this protocol depends on the version of MSMQ and Directory Service.

    1. If MSMQ version 1.0 Directory Service is running, then the MQDS itself informs the QM via MSMQ: Directory Service Change Notification Protocol, as shown in arrow 2.a.

    2. If Active Directory is used, then the QM on System 1 informs the QM on System 3 via MSMQ: Directory Service Change Notification Protocol, as shown in arrow 2.b.

2.2.6 Transaction and Routing Profile

This profile describes MSMQ's routing and transactional messaging. An application is sending a transactional message that is to be received by an application on an independent client on a remote **site**. The Initiating Application only knows the path name of the public queue. All the systems in this scenario are also aware of the **distributed transaction coordinators (DTC)** as well as the MSMQ

Directory Service implemented by Active Directory and the domain topology as specified in [MS-ADTS]. Following are the set of actions that take place to implement this scenario.

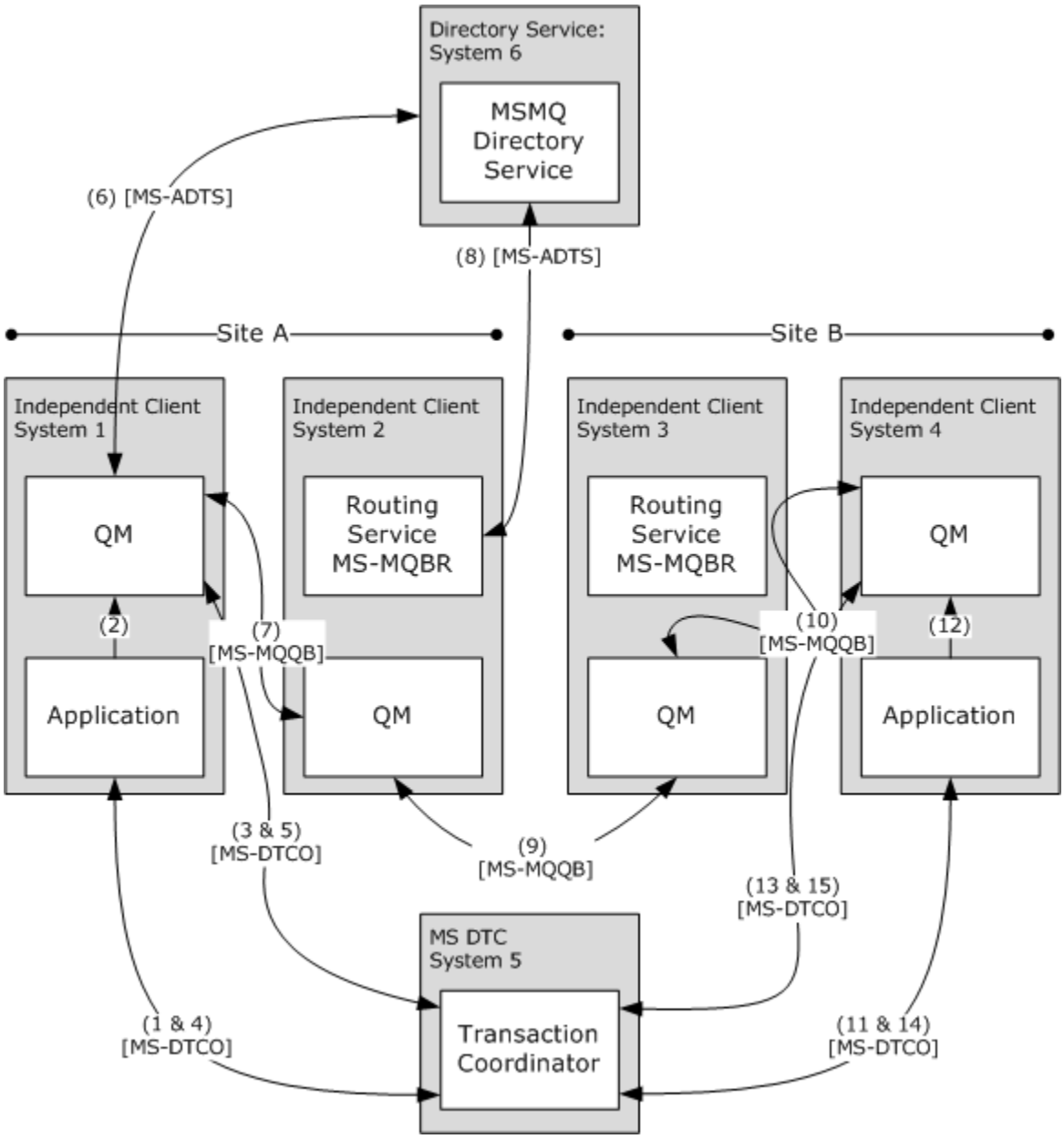

Figure 7: MSMQ routing sequence

1. The initiating application on System 1 needs to send a transactional message to a public **transactional queue** hosted on System 4. Transactionality is a property of the message; however, MSMQ uses transactional queue to send and receive transactional messages. In order to do so, the application creates a **transaction**; it interacts with a DTC, in this case on System 5, as specified in [MS-DTCO].

2. The application then sends messages to its local QM with the information the QM needs to enlist in the transaction as required by the MSDTC Connection Manager: OleTx Transaction Protocol.

3. The QM then enlists with the DTC; this communication happens over the MSDTC Connection Manager: OleTx Transaction Protocol.

4. Once the application is done sending messages to its QM, it commits the transaction using the protocol functionality as specified in [MS-DTCO].

5. The MS DTC on System 5 uses the two-phase commit protocol, as specified in [MS-DTCO], to commit the transaction. After all participants have committed, the QM on System 1 can start sending messages.

6. After the transaction is committed, the QM on System 1 can attempt to send messages, in order to do so it needs to query the MQDS to resolve the path name into a destination address. See [MS-ADTS] for more information about the LDAP protocol used to perform this query.

7. Because the destination queue is on a remote site, the QM on System 1 sends the message to the Routing Server in its site, in this case System 2. The QM on System 1 obtains information about System 2 by querying the MQDS. The message transfer between QM and Routing Server is specified in [MS-MQQB].

8. The Routing Server queries the MQDS, as specified in [MS-ADTS], for **site link** costs in order to determine the next hop for the message. Based on the Site Link costs, the routing algorithm determines the next hop. See [MS-MQBR] for more details.

9. The message is sent to the next routing server. For more information on the communication between the two Routing Servers, see [MS-MQQB]. In this application profile the Routing Server on System 3 is located in the same site as the Independent client on System 4 that contains the destination queue; hence the Routing Server on System 3 does not need to re-route the message.

10. The message is directly sent from the Routing Server on System 3 to the QM on System 4 over Message Queuing (MSMQ): Message Queuing Binary Protocol.

11. For the Receiving application to receive the transactional messages, it has to start another transaction with the DTC, in this case on System 7; see [MS-DTCO] for more information.

12. Under the new transaction the Application can now access the transactional messages from the QM on System 4.

13. Once again, the QM on System 4 enlists in this transaction; see [MS-DTCO] for the protocol communication details.

14. Once the receiving application is done receiving messages it commits the transaction; see [MS-DTCO] for more details.

15. The DTC uses the two phase commit protocol, as specified in [MS-DTCO], to commit all participants to this transaction. The QM commits the transaction.

2.2.6.1 Key Roles and Concepts

This application profile details the routing server as well as the DTC role in MSMQ transactional messages. These roles and related concepts are described in the following sections.

2.2.6.1.1 Routing Server

The routing server is an MSMQ installation with a QM that supports routing of messages between sites. The routing server stores the messages and then forwards them to their destinations or to alternate hops.

2.2.6.1.2 Distributed Transaction Coordinator

The Microsoft Distributed Transaction Coordinator (DTC) is a distributed transaction facility for Windows platforms. For more information, see [MS-DTCO].

2.3 MSMQ Directory Service

Microsoft Message Queuing, when operating in domain mode, relies on a directory service for publication, location, and manipulation of queues, and for message routing. The directory service is also used for storing security information.

The implementation of the directory service has evolved over MSMQ versions. In MSMQ version 1.0, the MQDS is implemented as a replicated directory service distributed across a set of sites. One site is designated as the **Primary Enterprise Controller (PEC)** and zero or more other sites are **Primary Site Controllers (PSC)**. The PEC acts as a PSC for its site. Each PSC can have one or more **Backup Site Controllers (BSC)**. Every site controller owns its individual site information and replicates this information to the PEC. Each site controller also contains a copy of information related to all objects in an enterprise; the PSC receives this information from the PEC. A top-level overview of the directory service implementation used by MSMQ 1.0 can be found in [MS-MQDS] section 3.1.1.1.

In MSMQ version 2.0, the MQDS is implemented using Active Directory (AD). AD was introduced into the operating system in Windows 2000 and MSMQ takes advantage of it in version 2.0.

3 Protocol Architecture

3.1 Protocol Overview

This section enumerates all of the protocols used by MSMQ, provides a brief description, and shows how those protocols are used by the various client and server roles.

The intent of the following sections is not to fully describe the characteristics of each protocol. The intent is to give a brief description of the protocol and describe where in the MSMQ architecture the protocol is used. See the protocol documentation associated with the protocol for full details of how each protocol works.

3.1.1 Message Queuing Binary Protocol

3.1.1.1 Overview

The Message Queuing Binary Protocol defines a mechanism for reliably transferring messages between two queue managers located on two different host computers. The protocol is stateful, wherein the client establishes a connection to a server and then sends a variety of packets to transfer messages. The protocol defines additional behaviors such as administration acknowledgments and message source journaling. The protocol uses the User Datagram Protocol (UDP) to determine server availability and Transmission Control Protocol/Sequenced Packet Exchange (TCP/SPX) to transport the data, but augments it with additional levels of acknowledgment that ensure the messages are reliably transferred irrespective of TCP/SPX connection failures, application failures, or node failures. For more information, see [MS-MQQB].

3.1.1.2 Relationship to Other Protocols

Message Queuing Binary Protocol is dependent upon direct Transmission Control Protocol/Internet Protocol (TCP/IP) or Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) to provide a reliable stream-oriented transport for messages. The protocol uses the User Datagram Protocol (UDP) or IPX to determine server availability. The protocol may rely upon LDAP, as specified in [MS-ADTS], or the Directory Service Protocol, as specified in [MS-MQDS], to look up persistently stored entries in Active Directory. There is no cross protocol timing, sequencing, or semantics related to reading and updating MSMQ information stored in AD.

3.1.2 Directory Service Protocol

3.1.2.1 Overview

The Message Queuing Directory Service Protocol defines a set of procedure calls that can be made between a client and an MSMQ Directory Server. The client uses these calls to access the Directory Service remotely. For example, a client may use this protocol to create queue objects in a directory.

This protocol is intended for use by message queuing clients and message queuing servers. The directory defined by this protocol is composed of eight types of directory objects, representing enterprises, sites, site links, machines, users, queues, connected networks, and deleted objects. The protocol provides methods to create, update, retrieve, and delete objects from the directory service using either the object name or the unique object **GUID** as a key to identify the object. For more information, see [MS-MQDS].

3.1.2.2 Relationship to Other Protocols

Message Queuing Directory Service Protocol is a Remote Procedure Call (RPC)-based protocol consisting of simple request-response exchanges. For every method request that the server receives, it executes the method and returns a completion. The client simply returns the completion status to the caller. This protocol depends upon RPC for its transport.

This protocol is deprecated. Implementers should use the Lightweight Directory Access Protocol (see [MS-ADTS] for more information) instead of this protocol except where compatibility requirements necessitate use of this protocol. This protocol relies on the Private Communications Technology Protocol for authentication and message security.

3.1.3 Directory Service Discovery Protocol

3.1.3.1 Overview

The Message Queuing Directory Service Discovery Protocol defines a mechanism used by queue managers or Routing Servers to obtain a current list of network accessible MSMQ servers running the MQDS implemented in MSMQ version 1. After an MQDS server has been located, the Message Queuing (MSMQ): Directory Service Protocol is used to obtain MSMQ configuration information such as the queues and machine names containing MSMQ installations. For more information about the directory service protocol, see [MS-MQSD].

3.1.3.2 Relationship to Other Protocols

Message Queuing (MSMQ): Directory Service Discovery Protocol depends on the User Datagram Protocol (UDP) and Internet Protocol (IP) or Internetwork Packet Exchange (IPX) protocols for sending discovery requests and receiving discovery replies. This protocol is not used by MSMQ Servers to discover other MSMQ Servers.

3.1.4 Queue Manager to Queue Manager Protocol

3.1.4.1 Overview

The Message Queuing Queue Manager to Queue Manager Protocol defines a mechanism for consuming messages from a remote queue. This protocol addresses scenarios where the queue manager may execute on a different node than the client applications. This scenario uses the Message Queuing Queue Manager to Queue Manager Protocol to insert and consume messages from the remote queue. Message Queuing Queue Manager to Queue Manager Protocol is an RPC-based protocol used by the queue manager and Dependent Client to read MSMQ messages from a remote QM. Operations that a client would perform via this protocol include reading an MSMQ message from a remote queue and purging (deleting) an MSMQ message from a remote queue. Reading a message may also imply deleting the message after it is read. For more information see [MS-MQQP].

3.1.4.2 Relationship to Other Protocols

Message Queuing Queue Manager to Queue Manager Protocol depends on Remote Procedure Call (RPC) for its transport. Some methods in this RPC interface operate on parameters that have been obtained from the qmcomm RPC interface as specified in [MS-MQMP]. The server implementation must access internal state from the Message Queuing (MSMQ): Queue Manager Client Protocol RPC interface, specifically for Queue and **cursor** handles. The Message Queuing (MSMQ): Queue Manager Client Protocol RPC interface creates Queue and Cursor handles that are required to be passed to the methods in this interface. This protocol is superseded by the Message Queuing

(MSMQ): Queue Manager Remote Read Protocol, which is used in MSMQ version 3 in Windows Server 2003, but is retained for use with downlevel clients and servers.

3.1.5 Queue Manager Client Protocol

3.1.5.1 Overview

The Message Queuing Queue Manager Client Protocol defines a mechanism to implement dependent client and supporting server functionality for MSMQ. Message Queuing Queue Manager Client Protocol is an RPC-based protocol that enables communication between a Dependent Client and a queue manager. Typically the QM is acting as a Supporting Server for the Dependent Client. However, this protocol is also used to initiate communication between a Dependent Client and a remote QM for the Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol when the Dependent Client performs a read from a remote queue. Operations that a client would perform via this protocol include managing private queues; opening and closing queue handles; enlisting, committing, and aborting internal transactions; enlisting the queue manager in external transactions; purging queues; creating cursors; sending messages; and reading messages. For more information, see [MS-MQMP].

3.1.5.2 Relationship to Other Protocols

Message Queuing (MSMQ): Queue Manager Client Protocol is dependent upon Remote Procedure Call (RPC) for its transport. This protocol collaborates with the functionality provided by the qm2qm RPC interface, as specified by [MS-MQQP].

3.1.6 Queue Manager Management Protocol

3.1.6.1 Overview

The Message Queuing Queue Manager Management Protocol defines a mechanism which is used by a Management Client for management operations on the queue manager, including obtaining the MSMQ installation version and monitoring the queues. This protocol can be used to obtain information from the queue manager such as queue properties, current queue state, auditing information, and so on. The protocol can also be used to perform certain actions on a machine and on a queue. For more information, see [MS-MQMR].

Message Queuing Queue Manager Management Protocol is a Remote Procedure Call (RPC)-based protocol. The server does not maintain client state information. The protocol operation is stateless.

3.1.6.2 Relationship to Other Protocols

Message Queuing Queue Manager Management Protocol is dependent upon Remote Procedure Call (RPC) over Transmission Control Protocol/Internet Protocol (TCP/IP) for its transport.

3.1.7 Directory Service Change Notification Protocol

3.1.7.1 Overview

The Message Queuing Directory Service Change Notification Protocol defines a mechanism used to notify a QM about changes on the objects that it owns. This protocol can be called either by MQDS implemented in MSMQ version 1 or by a different QM. Operations that can be performed using this protocol include notifying a QM that a queue object has been created, changed, or deleted; and notifying a QM that a machine object has been changed. For more information, see [MS-MQCN].

3.1.7.2 Relationship to Other Protocols

Message Queuing Directory Service Change Notification Protocol is a queued protocol that uses MSMQ message queuing itself as its transport protocol.

3.1.8 Queue Manager Remote Read Protocol

3.1.8.1 Overview

The Queue Manager Remote Read Protocol is a Remote Procedure Call (RPC)-based protocol that defines a mechanism which is used by the queue manager (QM) to read MSMQ messages from a remote QM. In addition, it allows MSMQ clients to reject a message from a queue, move a message between queues, and purge all messages from a queue. Queue Manager Remote Read Protocol is introduced in MSMQ version 3 on Windows Server 2003 and replaces Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol. Queue Manager Remote Read Protocol is a more secure and efficient protocol than Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol, and is enhanced in MSMQ version 4. For more information, see [MS-MQRR].

3.1.8.2 Relationship to Other Protocols

Queue Manager Remote Read Protocol is dependent upon Remote Procedure Call (RPC) for its transport. The protocol functionality is a superset of the functionality provided by Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol. Implementers should choose this protocol over MSMQ: Queue Manager to Queue Manager Protocol except where compatibility requirements necessitate use of MSMQ: Queue Manager to Queue Manager Protocol.

3.2 Protocol Usage

This section identifies how the MSMQ clients and servers use the protocols. In order to understand how MSMQ protocols operate, it is necessary to understand and define the various client and server roles for MSMQ. The following sections describe client and server capabilities of MSMQ.

3.2.1 MSMQ Clients

The MSMQ Client roles are noted in the following list. These roles are not dependent on the underlying operating system; they can be installed on both client and server versions of an operating system. These client roles are defined in the glossary.

- Independent Client
- Dependent Client
- Management Client

3.2.2 MSMQ Server

The MSMQ Server roles are noted in the following list. These roles are server-specific and need to be installed and functioning in a server operating system. The server roles are defined in the [MS-MQMA] and [MS-MQMQ] glossaries.

- **MSMQ Routing Server**
- MSMQ Directory Service
- MSMQ Supporting Server

- **Down-Level Service**

3.2.3 MSMQ Version 1.0 and MSMQ Version 2.0 Protocol Architecture

The following diagram shows the different MSMQ Client and Server roles and the protocols that interact between them. The following architecture diagram is valid for MSMQ Version 1.0 and 2.0.

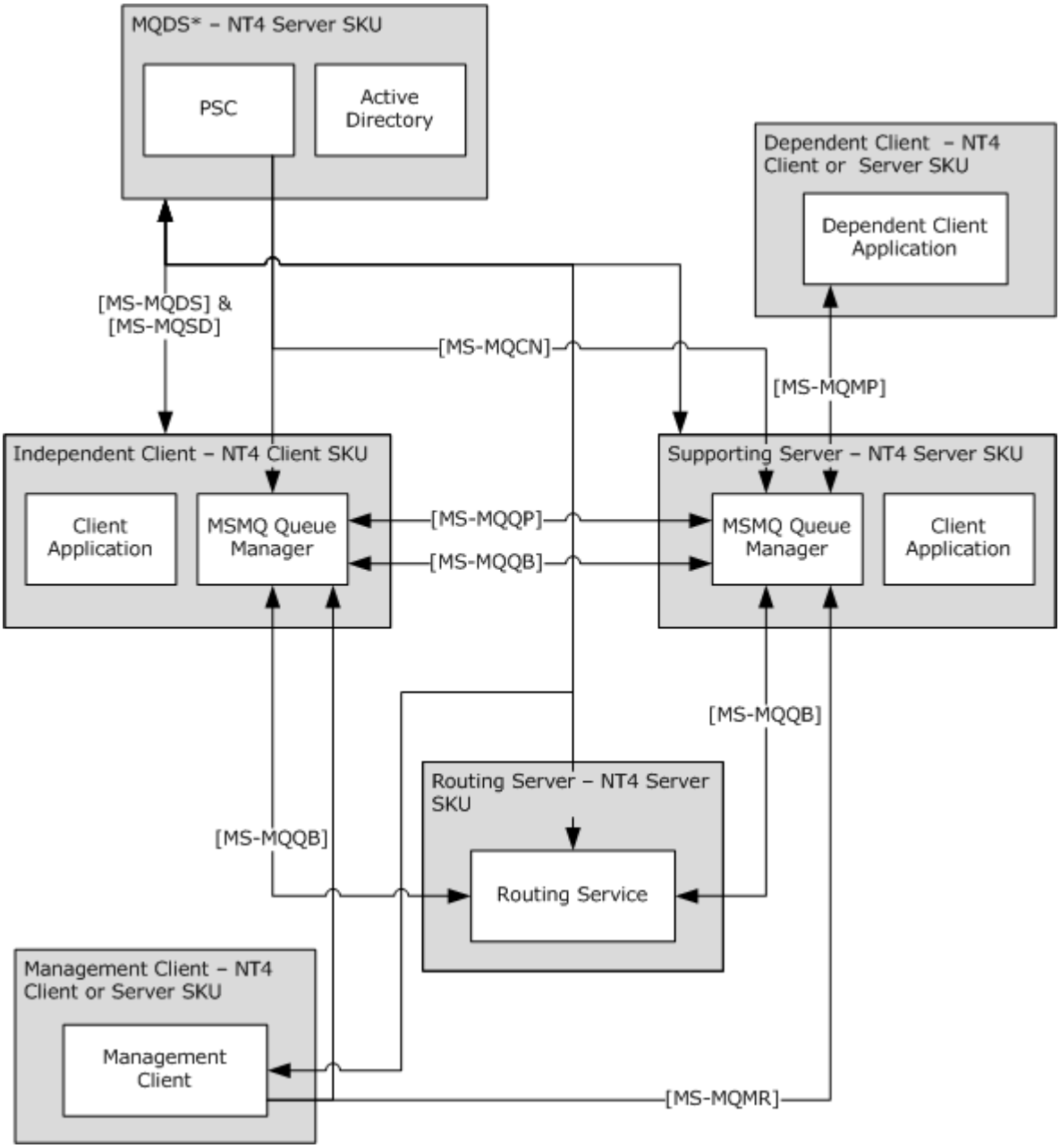

Figure 8: MSMQ client and server roles and protocols

Note MQDS is shown as being implemented either by MSMQ version 1.0 PSC or by Active Directory. In both MSMQ version 1.0 and MSMQ version 2.0, all MSMQ components interacted with MQDS over Message Queuing (MSMQ): Directory Service Protocol. If the DS is implemented using Active Directory, then MSMQ needs to be installed on an Active Directory domain controller; MSMQ relays the MQDS interaction over Message Queuing (MSMQ): Directory Service Protocol to Active Directory locally.

3.2.4 MSMQ Version 3.0 and Version 4.0 Protocol Architecture

The following describes the diagrams for both MSMQ version 3.0 and version 4.0 architecture. There are two major protocol changes. First, MSMQ can directly communicate with Active Directory domain controllers via LDAP implementation, as specified in [MS-ADTS]. Secondly, if two QMs are on Windows Server 2003, or any SKU that is released later, the Message Queuing (MSMQ): Queue Manager Remote Read Protocol protocol is used for remote read and receives instead of Message Queuing (MSMQ): Queue Manager to Queue Manager Protocol.

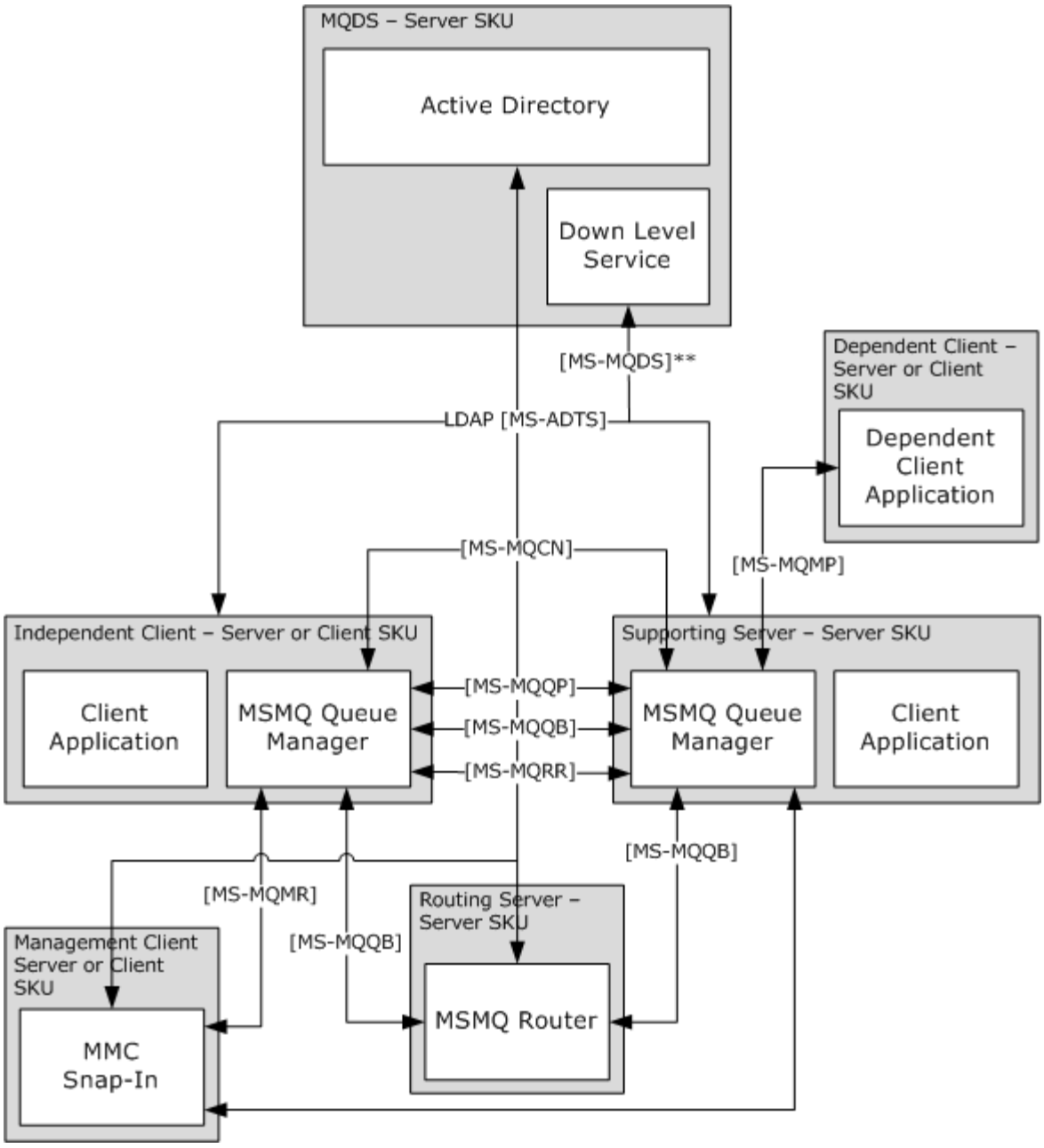

Figure 9: Queue Manager Remote Read Protocol used in place of Queue Manager to Queue Manager Protocol

Note The diagram shows Queue Manager Remote Read Protocol used in place of Queue Manager to Queue Manager Protocol. This only happens if the both QMs are using one of the following operating systems: Windows Server 2003 SKU, Windows Vista SKU or Windows Server 2008 SKU.

Note In this version of the architecture, downlevel clients interact with the MQDS over Message Queuing (MSMQ): Directory Service Protocol. This interaction happens over an MSMQ downlevel

service which acts as a local relay between Message Queuing (MSMQ): Directory Service Protocol and Active Directory.

4 Index

A

Applicability
Architecture
  product
  protocol

C

Client
  dependent - concepts
  dependent - defined
  dependent - domain mode
  dependent - key roles
  dependent - MSMQ directory service
  dependent - profile
  dependent - supporting server
  independent - concepts
  independent - defined
  independent - key roles
  independent - profile
  independent - workgroup mode
  management - defined
  management - query - concepts
  management - query - key roles
  management - query - profile
  management - update - profile
  MSMQ - usage
Concepts
  dependent client
  independent client
  management client query
  routing
  transaction

D

Dependent client
  concepts
  defined
  domain mode
  key roles
  MSMQ directory service
  profile
  supporting server
Directory Service Change Notification Protocol
Directory Service Discovery Protocol
Directory Service Protocol [MS-MQDS]
Distributed Transaction Coordinator
Domain mode - dependent client

G

Glossary

I

Independent client
  concepts
  defined
  key roles
  profile
  workgroup mode
Informative references
Introduction
Introduction - product architecture

K

Key roles
  dependent client
  independent client
  management client query
  routing
  transaction

M

Management client
  defined
  query - concepts
  query - key roles
  query - profile
  update - profile
Message Queuing Binary Protocol
MSMQ
  clients - usage
  directory service - dependent client
  directory service - overview
  server - usage
  version 1 - usage
  version 2 - usage
  version 3- usage
  version 4 - usage

N

Normative references

O

Overview
  Directory Service Change Notification Protocol
  Directory Service Protocol
  Directory Service Protocol [MS-MQDS]
  Message Queuing Binary Protocol
  protocol architectures
  Queue Manager Client Protocol
  Queue Manager Management Protocol
  Queue Manager Remote Read Protocol
  Queue Manager to Queue Manager Protocol

P

Product architecture - introduction
Profiles
  dependent client
  independent client

management client query
management client update
overview
routing
transaction

Q

Queue Manager Client Protocol
Queue Manager Management Protocol
Queue Manager Remote Read Protocol
Queue Manager to Queue Manager Protocol

R

References
informative
normative
overview
Relationship to other protocols
Directory Service Change Notification Protocol
Directory Service Protocol (section 3.1.2.2, section 3.1.3.2)
Message Queuing Binary Protocol
overview
Queue Manager Client Protocol
Queue Manager Management Protocol
Queue Manager Remote Read Protocol
Queue Manager to Queue Manager Protocol
Routing
concepts
key roles
profile
server - defined

S

Server
MSMQ - usage
routing - defined
supporting - dependent client
Supporting server - dependent client

T

Transaction
concepts
Distributed Coordinator
key roles
profile

U

Usage
MSMQ clients
MSMQ server
MSMQ version 1
MSMQ version 2
MSMQ version 3
MSMQ version 4
protocols

V

Version 1 - MSMQ
usage
Version 2 - MSMQ - usage
Version 3 - MSMQ - usage
Version 4 - MSMQ - usage

W

Workgroup mode - independent client