

# [MS-MQCN]: Message Queuing (MSMQ): Directory Service Change Notification Protocol Specification

---

## Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft [Open Specification Promise](#) or the [Community Promise](#). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications do 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. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

## Revision Summary

Date	Revision History	Revision Class	Comments
07/20/2007	0.1	Major	MCPP Milestone 5 Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.2.1	Editorial	Revised and edited the technical content.
11/30/2007	0.2.2	Editorial	Revised and edited the technical content.
01/25/2008	0.2.3	Editorial	Revised and edited the technical content.
03/14/2008	0.2.4	Editorial	Revised and edited the technical content.
05/16/2008	1.0	Major	Updated and revised the technical content.
06/20/2008	1.1	Minor	Updated the technical content.
07/25/2008	1.1.1	Editorial	Revised and edited the technical content.
08/29/2008	2.0	Major	Updated and revised the technical content.
10/24/2008	3.0	Major	Updated and revised the technical content.
12/05/2008	4.0	Major	Updated and revised the technical content.
01/16/2009	4.1	Minor	Updated the technical content.
02/27/2009	5.0	Major	Updated and revised the technical content.
04/10/2009	5.1	Minor	Updated the technical content.
05/22/2009	5.2	Minor	Updated the technical content.
07/02/2009	5.3	Minor	Updated the technical content.
08/14/2009	5.3.1	Editorial	Revised and edited the technical content.
09/25/2009	5.4	Minor	Updated the technical content.
11/06/2009	5.4.1	Editorial	Revised and edited the technical content.
12/18/2009	5.4.2	Editorial	Revised and edited the technical content.
01/29/2010	5.5	Minor	Updated the technical content.
03/12/2010	5.5.1	Editorial	Revised and edited the technical content.
04/23/2010	5.5.2	Editorial	Revised and edited the technical content.
06/04/2010	5.5.3	Editorial	Revised and edited the technical content.
07/16/2010	5.5.3	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
08/27/2010	6.0	Major	Significantly changed the technical content.
10/08/2010	7.0	Major	Significantly changed the technical content.
11/19/2010	7.1	Minor	Clarified the meaning of the technical content.
01/07/2011	7.2	Minor	Clarified the meaning of the technical content.
02/11/2011	8.0	Major	Significantly changed the technical content.
03/25/2011	9.0	Major	Significantly changed the technical content.
05/06/2011	10.0	Major	Significantly changed the technical content.
06/17/2011	10.1	Minor	Clarified the meaning of the technical content.

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>6</b>
1.1	Glossary .....	6
1.2	References.....	7
1.2.1	Normative References.....	7
1.2.2	Informative References .....	7
1.3	Overview .....	8
1.4	Relationship to Other Protocols.....	8
1.5	Prerequisites/Preconditions .....	8
1.6	Applicability Statement.....	9
1.7	Versioning and Capability Negotiation.....	9
1.8	Vendor-Extensible Fields.....	9
1.9	Standards Assignments .....	9
<b>2</b>	<b>Messages.....</b>	<b>10</b>
2.1	Transport.....	10
2.2	Message Syntax .....	10
2.2.1	GUID .....	10
2.2.2	PROPID .....	10
2.2.2.1	PROPID Constants .....	10
2.2.3	PROPVVALUE .....	12
2.2.3.1	VT_I2.....	12
2.2.3.2	VT_I4.....	12
2.2.3.3	VT_BOOL.....	12
2.2.3.4	VT_I1.....	12
2.2.3.5	VT_UI1.....	13
2.2.3.6	VT_UI2.....	13
2.2.3.7	VT_UI4.....	13
2.2.3.8	VT_I8.....	13
2.2.3.9	VT_UI8.....	14
2.2.3.10	VT_LPWSTR .....	14
2.2.3.11	VT_BLOB .....	14
2.2.3.12	VT_CLSID .....	15
2.2.3.13	VT_UI4   VT_VECTOR.....	15
2.2.3.14	VT_CLSID   VT_VECTOR .....	15
2.2.3.15	VT_LPWSTR   VT_VECTOR .....	15
2.2.4	Change Notification Message .....	16
2.2.5	Notification Body .....	17
2.2.6	Notification Update.....	18
2.3	Directory Service Schema Elements .....	20
<b>3</b>	<b>Protocol Details.....</b>	<b>21</b>
3.1	Common Details .....	21
3.1.1	Abstract Data Model .....	21
3.1.1.1	Shared Data Elements.....	21
3.1.1.2	Data Elements Mapping.....	21
3.1.1.2.1	Queue Manager Attributes Mapping .....	21
3.1.1.3	Queue Attributes Mapping.....	22
3.1.2	Timers .....	22
3.1.3	Initialization .....	22
3.1.4	Higher-Layer Triggered Events.....	22

3.1.5	Processing Events and Sequencing Rules .....	23
3.1.6	Timer Events .....	23
3.1.7	Other Local Events .....	23
3.2	MQCN Server Details .....	23
3.2.1	Abstract Data Model .....	23
3.2.1.1	Shared Data Elements .....	23
3.2.1.2	Local Data Elements .....	23
3.2.1.2.1	Change Notification Queue .....	23
3.2.2	Timers .....	24
3.2.3	Initialization .....	24
3.2.4	Higher-Layer Triggered Events .....	24
3.2.5	Processing Events and Sequencing Rules .....	24
3.2.5.1	Processing a Change Notification Message version 0x01 .....	25
3.2.5.2	Processing a Change Notification Message version 0x02 .....	27
3.2.6	Timer Events .....	28
3.2.7	Other Local Events .....	28
3.3	MQCN Client Details .....	28
3.3.1	Abstract Data Model .....	28
3.3.1.1	Shared Data Elements .....	29
3.3.2	Timers .....	29
3.3.3	Initialization .....	29
3.3.4	Higher-Layer Triggered Events .....	29
3.3.4.1	Send Change Notification .....	29
3.3.5	Processing Events and Sequencing Rules .....	30
3.3.5.1	Preparing a Change Notification Message Version 0x02 .....	30
3.3.5.2	Preparing a Change Notification Message Version 0x01 .....	30
3.3.5.3	Sending a Change Notification Message .....	33
3.3.6	Timer Events .....	34
3.3.7	Other Local Events .....	34
<b>4</b>	<b>Protocol Examples .....</b>	<b>35</b>
4.1	Management Client Update Profile .....	35
<b>5</b>	<b>Security .....</b>	<b>37</b>
5.1	Security Considerations for Implementers .....	37
5.2	Index of Security Parameters .....	37
<b>6</b>	<b>Appendix A: Product Behavior .....</b>	<b>38</b>
<b>7</b>	<b>Change Tracking .....</b>	<b>41</b>
<b>8</b>	<b>Index .....</b>	<b>43</b>

# 1 Introduction

This document specifies the Message Queuing (MSMQ): Directory Service Change Notification Protocol. **Queue managers** own and store objects that are also stored in the **directory service**. When an application makes changes to an object in the directory service, the MSMQ: Directory Service Change Notification Protocol is used by the **MSMQ Directory Service** or by the queue manager to notify the queue manager that owns the object that those changes have occurred. The types of notifications that can be performed by using this protocol include notifying a queue manager that a **queue** object has been created, changed, or deleted; and notifying a queue manager that its machine object has been changed. The MSMQ: Directory Service Change Notification Protocol is a queued protocol that uses **Microsoft Message Queuing (MSMQ)** as its transport infrastructure to send notifications wrapped within MSMQ **messages**.

## 1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

**Active Directory**  
**Augmented Backus-Naur Form (ABNF)**  
**domain controller (DC)**  
**globally unique identifier (GUID)**  
**little-endian**  
**Unicode character**  
**Unicode string**

The following terms are defined in [\[MS-MQMQ\]](#):

**connected network**  
**directory service**  
**enterprise**  
**format name**  
**message**  
**message body**  
**message property**  
**message queue**  
**Microsoft Message Queuing (MSMQ)**  
**MSMQ Directory Service**  
**MSMQ management server**  
**MSMQ queue manager**  
**MSMQ routing link**  
**MSMQ routing server**  
**MSMQ site**  
**private queue**  
**public queue**  
**queue**  
**queue manager**  
**workgroup mode**

The following terms are specific to this document:

**notification queue:** A private **Microsoft Message Queuing (MSMQ) queue** to which notifications are sent and from which notifications are received.

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 1.2 References

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

### 1.2.1 Normative 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](mailto: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.

[MS-ADTS] Microsoft Corporation, "[Active Directory Technical Specification](#)".

[MS-DTYP] Microsoft Corporation, "[Windows Data Types](#)".

[MS-MQBR] Microsoft Corporation, "[Message Queuing \(MSMQ\): Binary Reliable Message Routing Algorithm](#)".

[MS-MQDMPR] Microsoft Corporation, "[Message Queuing \(MSMQ\): Common Data Model and Processing Rules](#)".

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

[MS-MQDSSM] Microsoft Corporation, "[Message Queuing \(MSMQ\): Directory Service Schema Mapping](#)".

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

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

[MS-MQSO] Microsoft Corporation, "[Message Queuing System Overview](#)".

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, <http://www.ietf.org/rfc/rfc1321.txt>

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/rfc/rfc2119.txt>

[RFC3110] Eastlake III, D., "RSA/SHA-1 SIGs and RSA KEYS in the Domain Name System (DNS)", RFC 3110, May 2001, <http://www.ietf.org/rfc/rfc3110.txt>

### 1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

## 1.3 Overview

Microsoft Message Queuing (MSMQ) is a communications service that provides asynchronous and reliable message passing between client applications running on different hosts. In MSMQ, clients send application messages to queues and consume application messages from queues. The queue provides persistence of the messages, enabling them to survive across application restarts, and allowing the sending and receiving client applications to send and receive messages asynchronously from each other.

Queues are typically hosted by a communications service called a queue manager. By hosting the queue manager in a separate service from the client applications, applications can communicate (even if they never execute at the same time) by exchanging messages via a queue hosted by the queue manager.

Because MSMQ involves message passing between nodes, a directory service can be useful to MSMQ services in several ways. First, a directory service can provide network topology information that the MSMQ services can use to route messages between nodes. Second, a directory service can be used as a key distribution mechanism for security services used by MSMQ to secure messages and to authenticate clients. Third, a directory service can provide clients with discovery capabilities, allowing clients to discover the queues available within the network. Finally, a directory service can contain collections of directory objects representing **enterprises, sites, routing links**, machines, users, queues, **connected networks**, and deleted objects.

In MSMQ, queue and machine objects can be created, changed, and deleted in a directory service. As a result, the internal state of the queue manager that is the owner of these objects is left out of sync. In MSMQ, a queue manager is notified when one of its owned objects is changed in a directory service.

The MSMQ: Directory Service Change Notification Protocol defines a mechanism used by the MSMQ Directory Service or a queue manager to notify a queue manager of changes to its owned objects. The types of notifications that can be performed by using this protocol include notifying a queue manager that a queue object has been created, changed, or deleted; and notifying a queue manager that its machine object has been changed.

## 1.4 Relationship to Other Protocols

The MSMQ: Directory Service Change Notification Protocol is a queued protocol that uses the Microsoft Message Queuing (MSMQ): Message Queuing Binary Protocol [\[MS-MQOB\]](#) as its transport protocol.

The MSMQ: Directory Service Change Notification Protocol uses shared state and processing rules defined in [\[MS-MQDMPR\]](#).

## 1.5 Prerequisites/Preconditions

It is assumed that the protocol client has obtained the name of a server computer that supports this protocol and the name of the **notification queue** hosted on the server before this protocol is invoked. How a client acquires this information is not addressed in this specification and is typically part of the interaction between the client application and the queue manager API.

It is assumed that the protocol client has access to a private encryption key used to decrypt messages. A private key is typically stored in a secure location on the local client machine.



## 1.6 Applicability Statement

The server side of this protocol is applicable for implementation by a queue manager providing Microsoft Message Queuing (MSMQ) communication services to clients. The client side of this protocol is applicable for implementation by client libraries providing message queue managers to applications by directory service or by a queue manager delegating requests on behalf of a client. [<1>](#) This protocol is applicable to scenarios where a queue manager that is the owner of public queue objects or a machine object needs to be notified when these objects are changed in a directory service or on another queue manager on behalf of a client. [<2>](#) This protocol is not applicable to objects that are not stored in a directory service, such as private queues. This protocol is not applicable for distributed applications that require notification messages within a predefined amount of time. Notification messages are sent and, once received, the destination queue manager schedules act on them at specific time intervals. [<3>](#)

## 1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- Supported Transports: This protocol is implemented on top of MSMQ, as specified in section [2.1](#). It relies on MSMQ message sending and receiving mechanisms to send and receive notifications as messages.
- Capability Negotiation: This protocol has explicit capability negotiation that depends on the version of the notification message. There are two notification message versions. Version 1 is for messages sent by an MSMQ Directory Service, and version 2 is for messages sent by a queue manager. [<4>](#)

## 1.8 Vendor-Extensible Fields

This protocol does not define any vendor-extensible fields.

## 1.9 Standards Assignments

This section specifies standard parameters within the context of MSMQ.

Parameter	Value	Reference
NOTIFY_QUEUE_PRIVATE	PRIVATE=<Queue Manager GUID>\3	None.

## 2 Messages

### 2.1 Transport

Notifications are sent within messages that are transported as MSMQ messages over the Message Queuing (MSMQ): Message Queuing Binary Protocol [\[MS-MQQB\]](#).

### 2.2 Message Syntax

The following table summarizes the types and messages defined in this specification.

Message or type	Description
<a href="#">GUID</a>	<b>globally unique identifier (GUID).</b>
<a href="#">PROPID</a>	Property identifier.
<a href="#">PROPVALUE</a>	Property value.
<a href="#">Change Notification Message</a>	Payload of a notification message.
<a href="#">Notification Body</a>	Body of a notification. Used by a queue manager.
<a href="#">Notification Update</a>	Alternative body notification. Used by a directory service.

#### 2.2.1 GUID

This specification uses a globally unique identifier (GUID). This information is as specified in [\[MS-DTYP\]](#) section 2.3.2.

#### 2.2.2 PROPID

Notification messages may carry an array of property identifiers (a unique PROPID value). The associated property values are specified in a related array of property values.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Value																															

**Value (4 bytes):** A [ULONG](#) that specifies the identification of a property. This field MUST be filled with a valid property identifier, as specified in [\[MS-MQMQ\]](#) section 2.3.

##### 2.2.2.1 PROPID Constants

This section contains a list of [PROPID](#) constants that are specified in [\[MS-MQMQ\]](#).

PROPID Constant	Value
<a href="#">PROPID_Q_ADS_PATH</a>	126
<a href="#">PROPID_Q_AUTHENTICATE</a>	111

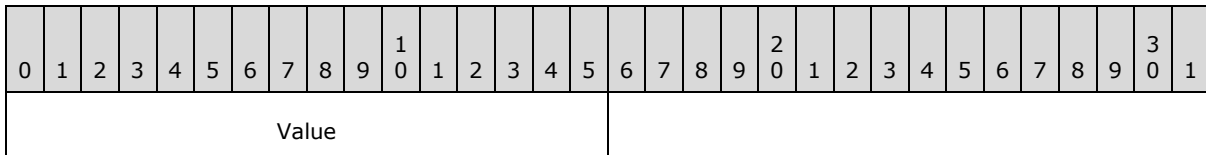
<b>PROPID Constant</b>	<b>Value</b>
<a href="#"><u>PROPID_Q_BASEPRIORITY</u></a>	106
<a href="#"><u>PROPID_Q_CREATE_TIME</u></a>	109
<a href="#"><u>PROPID_Q_INSTANCE</u></a>	101
<a href="#"><u>PROPID_Q_JOURNAL</u></a>	104
<a href="#"><u>PROPID_Q_JOURNAL_QUOTA</u></a>	107
<a href="#"><u>PROPID_Q_LABEL</u></a>	108
<a href="#"><u>PROPID_Q_MODIFY_TIME</u></a>	110
<a href="#"><u>PROPID_Q_MULTICAST_ADDRESS</u></a>	125
<a href="#"><u>PROPID_Q_PATHNAME</u></a>	103
<a href="#"><u>PROPID_Q_PRIV_LEVEL</u></a>	112
<a href="#"><u>PROPID_Q_SCOPE</u></a>	114
<a href="#"><u>PROPID_Q_QMID</u></a>	115
<a href="#"><u>PROPID_Q_QUOTA</u></a>	105
<a href="#"><u>PROPID_Q_SECURITY</u></a>	1101
<a href="#"><u>PROPID_Q_TRANSACTION</u></a>	113
<a href="#"><u>PROPID_Q_TYPE</u></a>	102
<a href="#"><u>PROPID_QM_CNS</u></a>	207
<a href="#"><u>PROPID_QM_FOREIGN</u></a>	219
<a href="#"><u>PROPID_QM_JOURNAL_QUOTA</u></a>	215
<a href="#"><u>PROPID_QM_SITE_ID</u></a>	201
<a href="#"><u>PROPID_QM_MACHINE_ID</u></a>	202
<a href="#"><u>PROPID_QM_ADDRESS</u></a>	206
<a href="#"><u>PROPID_QM_QUOTA</u></a>	214
<a href="#"><u>PROPID_QM_MACHINE_TYPE</u></a>	216
<a href="#"><u>PROPID_QM_MODIFY_TIME</u></a>	218
<a href="#"><u>PROPID_QM_OS</u></a>	220
<a href="#"><u>PROPID_QM_SECURITY</u></a>	1201
<a href="#"><u>PROPID_D_SCOPE</u></a>	1403
<a href="#"><u>PROPID_D_OBJTYPE</u></a>	1404

### 2.2.3 PROPVALUE

This section contains block diagrams for property values related to [PROPVARIANT](#) types, as specified in [\[MS-MQMQ\]](#) section 2.2.12. A PROPVARIANT type is determined by the [PROPID](#). [\[MS-MQMQ\]](#) section 2.3 specifies the PROPVARIANT type for each PROPID. All numeric values within the block diagrams MUST be formatted in **little-endian** byte order. Values in an array are simply concatenated with no padding.

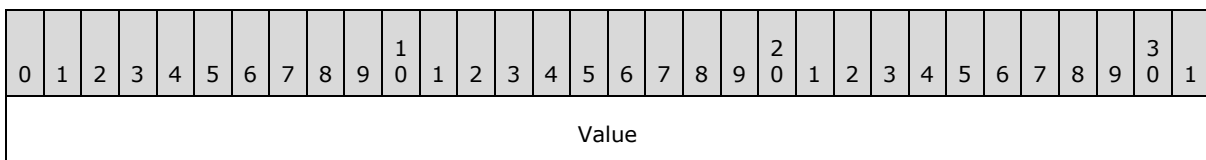
#### 2.2.3.1 VT\_I2

The value of a VT\_I2 type property MUST be a 2-byte signed integer. It MUST be formatted in little-endian byte order.



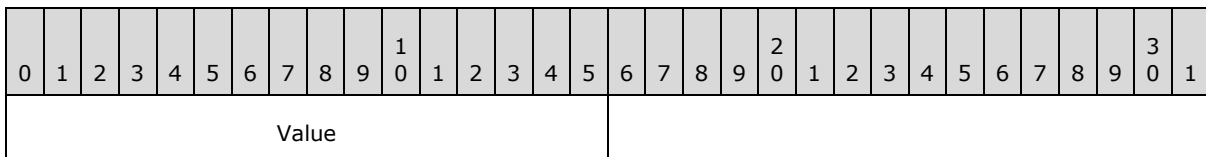
#### 2.2.3.2 VT\_I4

The value of a VT\_I4 type property MUST be a 4-byte signed integer. It MUST be formatted in little-endian byte order.



#### 2.2.3.3 VT\_BOOL

The value of a VT\_BOOL type property MUST be a 16-bit value. It MUST be formatted in little-endian byte order.

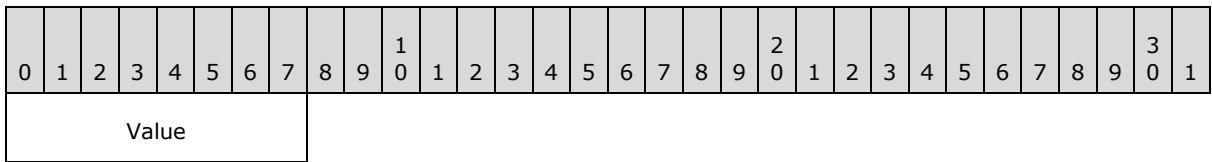


**Value (2 bytes):** The values MUST be defined as follows.

Value	Meaning
VARIANT_TRUE 0xFFFF	MUST indicate a Boolean value of true.
VARIANT_FALSE 0x0000	MUST indicate a Boolean value of false.

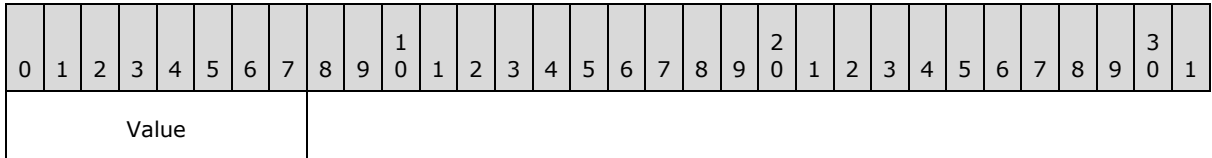
#### 2.2.3.4 VT\_I1

The value of a VT\_I1 type property MUST be a 1-byte signed integer.



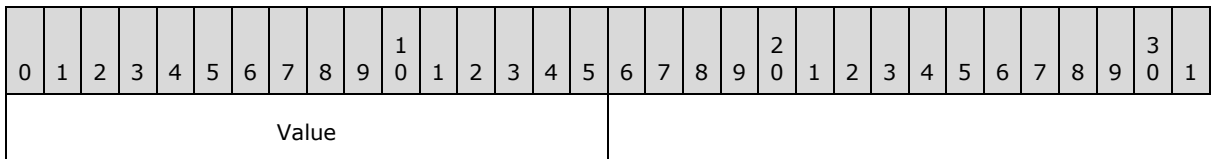
### 2.2.3.5 VT\_UI1

The value of a VT\_UI1 type property MUST be a 1-byte unsigned integer.



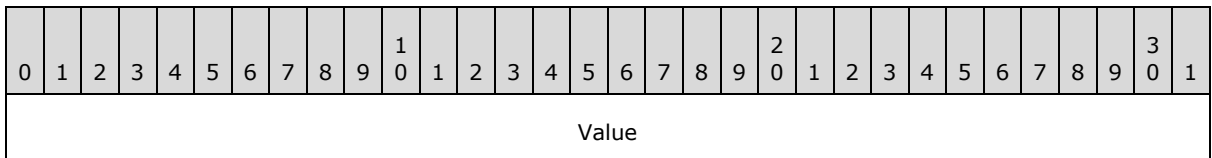
### 2.2.3.6 VT\_UI2

The value of a VT\_UI2 type property MUST be a 2-byte unsigned integer. It MUST be formatted in little-endian byte order.



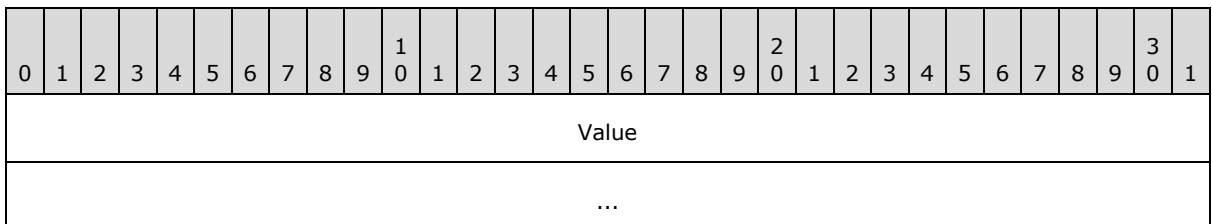
### 2.2.3.7 VT\_UI4

The value of a VT\_UI4 type property MUST be a 4-byte unsigned integer. It MUST be formatted in little-endian byte order.



### 2.2.3.8 VT\_I8

The value of a VT\_I8 type property MUST be an 8-byte signed integer. It MUST be formatted in little-endian byte order.



### 2.2.3.9 VT\_UI8

The value of a VT\_UI8 type property MUST be an 8-byte unsigned integer. It MUST be formatted in little-endian byte order.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Value																															
...																															

### 2.2.3.10 VT\_LPWSTR

The value of a VT\_LPWSTR type property MUST be a null-terminated string of 16-bit **Unicode characters**.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Unicode Characters (variable)																															
...																															
NullTerminator																															

**Unicode Characters (variable):** Contains a variable-length array of Unicode characters. Each array element MUST be set to a 16-bit Unicode character.

**NullTerminator (2 bytes):** Contains a Unicode character. This field MUST be set to a value of 0x0000.

### 2.2.3.11 VT\_BLOB

The value of a VT\_BLOB property MUST be a counted array of unsigned bytes.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Size																															
Value (variable)																															
...																															

**Size (4 bytes):** A **ULONG** that specifies the size of the array or vector of values. It MUST be formatted in little-endian byte order.

**Value (variable):** Contains an array of unsigned bytes. This field MUST be set to an array of length of *Size*, and each element MUST be an unsigned byte.

### 2.2.3.12 VT\_CLSID

The value of a VT\_CLSID type property MUST be a **GUID** ([\[MS-DTYP\]](#) section 2.3.2.2) value.

### 2.2.3.13 VT\_UI4 | VT\_VECTOR

The value of a VT\_UI4 | VT\_VECTOR type property MUST be an array of [VT\\_UI4](#) type property values.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Size																															
Value (variable)																															
...																															

**Size (4 bytes):** A [ULONG](#) that specifies the size of the array or vector of values. It MUST be formatted in little-endian byte order.

**Value (variable):** Value MUST be filled with an array of property values of VT\_UI4 type. The number of values MUST be equal to *Size*.

### 2.2.3.14 VT\_CLSID | VT\_VECTOR

The value of a VT\_CLSID | VT\_VECTOR type property MUST be an array of [VT\\_CLSID](#) type property values.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Size																															
Value (variable)																															
...																															

**Size (4 bytes):** A [ULONG](#) that specifies the size of the array or vector of values; MUST be formatted in little-endian byte order.

**Value (variable):** MUST be filled with an array of property values of VT\_CLSID type. The number of values MUST be equal to *Size*.

### 2.2.3.15 VT\_LPWSTR | VT\_VECTOR

The value of a VT\_LPWSTR | VT\_VECTOR type property MUST be an array of [VT\\_LPWSTR](#) type property values.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Size																															
Value (variable)																															
...																															

**Size (4 bytes):** A [ULONG](#) that specifies the size of the array or vector of values; MUST be formatted in little-endian byte order.

**Value (variable):** MUST be filled with an array of property values of VT\_LPWSTR type. The number of values MUST be equal to *Size*.

## 2.2.4 Change Notification Message

A Change Notification Message is used to encapsulate a single change [Notification Body](#) sent by a queue manager or several change [Notification Updates](#) sent by a directory service. A change notification represents a change on either a queue or a machine object within a directory service.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Version								NumberOfUpdateNotifications								Data (variable)																							
...																																							

**Version (1 byte):** A [UCHAR](#) containing the version of the notification message. This field MUST be set to the value 0x01 for messages sent by an MSMQ Directory Service and to the value 0x02 for messages sent by an **MSMQ queue manager**.[<5>](#)

Value	Meaning
0x01	Message sent by an MSMQ Directory Service.
0x02	Message sent by an MSMQ queue manager.

**NumberOfUpdateNotifications (1 byte):** A [UCHAR](#) that contains the number of update notifications. This field MUST be set to the number of update notifications contained in the field *Data*. If *Version* is set to 0x02, this field MUST be set to 1.

**Data (variable):** A variable-length [UCHAR](#) array. The content of this field varies depending on the **Version** field. If **Version** is set to 0x01 it contains a sequence of Notification Updates (see section [2.2.6](#)). If **Version** is set to 0x02 it contains a single Notification Body (section [2.2.5](#)). There is no padding between Notification Updates.



## 2.2.5 Notification Body

A Notification Body is used to encapsulate a change notification that indicates a change either on a queue or a machine object within a Directory Service. The change is expressed as a notification event that indicates one of the following events: creation of a queue object, change on a queue object, deletion of a queue object, or change on a machine object. A Notification Body is used by a queue manager to send a change notification message to the queue manager that is the owner of the modified object. This format does not include the changed data. The recipient queue manager **MUST** read the data from the directory service, as specified in section [3.2.5.2](#); therefore, queue managers do not have to be trusted.

<Notification>

<Event>**Event**</Event>

<ObjectGuid>**ObjectGuid**</ObjectGuid>

<DomainController>**DirectoryServiceServer**</DomainController>

</Notification>

### Event

Specifies the notification event for a queue or a machine object on which changes were made within a **domain controller (DC)**. This **MUST** be set to 1, 2, 3, or 4. A value of 1 indicates creation of a queue object. A value of 2 indicates change on a queue object; a value of 3 indicates deletion of a queue object. A value of 4 indicates change on a machine object.

### ObjectGuid

Specifies the object that was modified within the DC. This **MUST** be set to the [GUID](#) of the modified object that **MUST** be either a queue or a machine object.

### DirectoryServiceServer

Specifies the name of the Directory Service server through which the queue or the machine object was modified in the directory.

Notification body **MUST** be set to a **Unicode string** in the format specified by the following **Augmented Backus-Naur Form (ABNF)** rule.

```
NotificationBody = "<Notification>"
                  "<Event>"event"</Event>"
                  "<ObjectGuid>"object-guid"</ObjectGuid >"
                  "<DomainController>"
                    DirectoryServiceServer
                  "</DomainController>"
                  "</Notification>"

event             = "1" / "2" / "3" / "4"
                  ;where "1": create queue, "2": change queue,
                  ;"3": delete queue, "4": change machine

object-guid       = 8HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-" 4HEXDIG "-" 12HEXDIG
                  ; A GUID of the form XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

DirectoryServiceServer = 1*256(VCHAR)
                  ; A directory service name.
```

```

HEXDIG      = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
DIGIT       = %x30-39
              ; 0-9
VCHAR       = %x21-7E

```

## 2.2.6 Notification Update

Notification Update is used to encapsulate a change notification that indicates a change either on a queue or a machine object within a Directory Service. The change is expressed as a notification event that indicates one of the following events: creation of a queue object, change on a queue object, deletion of a queue object, or change on a machine object. Notification Update is used by a directory service to send change notification messages to the queue manager that is the owner of the modified objects.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31									
Command								UseGuid								PathName (variable)																								
...																																								
GuidIdentifier (optional)																																								
...																																								
...																																								
...																																								
GuidMasterId																																								
...																																								
...																																								
...																																								
Reserved																																								
...																																								
...																																								
...																																								
...																																								

...	
NumberOfProperties	PropertyId (variable)
...	
PropertyValue (variable)	
...	

**Command (1 byte):** A [UCHAR](#) that indicates the change notification operation, which can be create, change, or delete. This field **MUST** be set to 0x00, 0x01, or 0x02, where:

Value	Meaning
0x00	Indicates creation.
0x01	Indicates change.
0x02	Indicates deletion.

**UseGuid (1 byte):** A [UCHAR](#) that indicates if a Notification Update uses the *PathName* or the *GuidIdentifier* field. This field **MUST** be set to 0x01 to indicate that *GuidIdentifier* is being used. Otherwise, this field **MUST** be set to 0x00 to indicate that *PathName* is being used.

Value	Meaning
0x00	<i>PathName</i> is being used.
0x01	<i>GuidIdentifier</i> is being used.

**PathName (variable):** Indicates the path of the object that was modified in the directory service. **MUST** be a null-terminated Unicode string, and **MUST** follow the rules defined in [PROPID\\_Q\\_PATHNAME](#), as specified in [\[MS-MQMQ\]](#) section 2.3.1.3. This field **MUST NOT** be present if *UseGuid* is set to 0x01. Instead, *GuidIdentifier* **MUST** be set with a value.

**GuidIdentifier (16 bytes):** Indicates the [GUID](#) of the object that was modified in the directory service. This field **MUST** be present if *UseGuid* is set to 0x01.

**GuidMasterId (16 bytes):** Indicates the **GUID** of the directory server that originated the change Notification Update.

**Reserved (24 bytes):** A 24-byte array of [UCHAR](#) reserved for future use. **MAY** be set to 0 by the client, and **MUST** be ignored by the server.

**NumberOfProperties (1 byte):** A [UCHAR](#) that indicates the number of properties that are included as part of this Notification Update.

**PropertyId (variable):** An array of [PROPID](#) that contains a collection of property IDs that are part of this Notification Update. These properties refer either to informative data or to changed data related to the changed object. The number of elements **MUST** be equal to the value of *NumberOfProperties*.

**PropertyValue (variable):** Contains a collection of property values that are part of this Notification Update. These instances contain either informative data or modified data related to the changed object. The number of property values MUST be equal to the value of *NumberOfProperties*. The format of each property value depends on the variant type, which is determined by the corresponding property ID. Section [2.2.3](#) lists the formats for the variant types. Sections [2.2.3.1](#) through [2.2.3.15](#) define the types for each property ID.

## 2.3 Directory Service Schema Elements

This protocol uses ADM elements specified in section [3.1.1](#). A subset of these elements can be published in a directory. This protocol SHOULD [<6>](#) access the directory using the algorithm specified in [\[MS-MQDSSM\]](#) and using LDAP [\[MS-ADTS\]](#). The Directory Service schema elements for ADM elements published in the directory are defined in [\[MS-MQDSSM\]](#) section 2.4. [<7>](#)

## 3 Protocol Details

### 3.1 Common Details

#### 3.1.1 Abstract Data Model

This conceptual model is common to both servers and clients of this protocol. The abstract data model, which is specific to either the server or client side of this protocol, is described in the MQCN Server Details ([3.2](#)) and MQCN Client Details ([3.3](#)) sections, respectively.

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The abstract data model for this protocol comprises elements that are private to this protocol and others that are shared between multiple MSMQ protocols that are colocated at a common queue manager. The shared abstract data model is defined in [\[MS-MQDMPR\]](#) section 3.1.1, and the relationship between this protocol, a queue manager, and other protocols that share a common queue manager is described in [\[MS-MQSO\]](#).

Section [3.1.1.1](#) references the elements from the shared data model that are manipulated by this protocol.

Section [3.1.1.2](#) describes the mapping between property identifiers and attributes of abstract data models elements.

##### 3.1.1.1 Shared Data Elements

The server and client side of this protocol manipulate the following abstract data model elements from the shared abstract data model defined in [\[MS-MQDMPR\]](#) section 3.1.1.

**QueueManager:** A data element as defined in [\[MS-MQDMPR\]](#) section 3.1.1.1.

**Queue:** A data element as defined in [\[MS-MQDMPR\]](#) section 3.1.1.2.

**Message:** A data element as defined in [\[MS-MQDMPR\]](#) section 3.1.1.12.

##### 3.1.1.2 Data Elements Mapping

This section describes the mapping between property identifiers and attributes of abstract data models elements.

###### 3.1.1.2.1 Queue Manager Attributes Mapping

The following section specifies the mapping between property identifiers and attributes of a queue manager data element.

**QueueManager.Identifier:** Defined in [PROPID\\_QM\\_MACHINE\\_ID](#).

**QueueManager.ForeignSystem:** Defined in [PROPID\\_QM\\_FOREIGN](#).

**QueueManager.QueueManagerQuota:** Defined in [PROPID\\_QM\\_QUOTA](#).

**QueueManager.JournalQuota:** Defined in [PROPID\\_QM\\_JOURNAL\\_QUOTA](#).

**QueueManager.Security:** Defined in [PROPID\\_QM\\_SECURITY](#).

**QueueManager.ConnectedNetworkIdentifierList:** Defined in [PROPID\\_QM\\_CNS](#).

**QueueManager.QueueManagerVersion:** Defined in [PROPID\\_QM\\_MACHINE\\_TYPE](#).

**QueueManager.OperatingSystemType:** Defined in [PROPID\\_QM\\_OS](#).

### 3.1.1.3 Queue Attributes Mapping

The following section specifies the mapping between property identifiers and attributes of a queue data element.

**Queue.Identifier:** Maps to [PROPID\\_Q\\_INSTANCE](#).

**Queue.Type:** Maps to [PROPID\\_Q\\_TYPE](#).

**Queue.BasePriority:** Maps to [PROPID\\_Q\\_BASEPRIORITY](#).

**Queue.Journaling:** Maps to [PROPID\\_Q\\_JOURNAL](#).

**Queue.Quota:** Maps to [PROPID\\_Q\\_QUOTA](#).

**Queue.JournalQuota:** Maps to [PROPID\\_Q\\_JOURNAL\\_QUOTA](#).

**Queue.CreateTime:** Maps to [PROPID\\_Q\\_CREATE\\_TIME](#).

**Queue.ModifyTime:** Maps to [PROPID\\_Q\\_MODIFY\\_TIME](#).

**Queue.Security:** Maps to [PROPID\\_Q\\_SECURITY](#).

**Queue.Pathname:** Maps to [PROPID\\_Q\\_PATHNAME](#).

**Queue.Label:** Maps to [PROPID\\_Q\\_LABEL](#).

**Queue.Authentication:** Maps to [PROPID\\_Q\\_AUTHENTICATE](#).

**Queue.PrivacyLevel:** Maps to [PROPID\\_Q\\_PRIV\\_LEVEL](#).

**Queue.Transactionnal:** Maps to [PROPID\\_Q\\_TRANSACTION](#).

**Queue.MulticastAddress:** Maps to [PROPID\\_Q\\_MULTICAST\\_ADDRESS](#).

**Queue.DirectoryPath:** Maps to [PROPID\\_Q\\_ADS\\_PATH](#).

**Queue.Scope:** Maps to [PROPID\\_Q\\_SCOPE](#).

### 3.1.2 Timers

There are no common timers.

### 3.1.3 Initialization

There is no common initialization.

### 3.1.4 Higher-Layer Triggered Events

There are no common higher-layer triggered events.

### 3.1.5 Processing Events and Sequencing Rules

There are no common message processing events and sequencing rules.

### 3.1.6 Timer Events

There are no common timer events.

### 3.1.7 Other Local Events

There are no other local events.

## 3.2 MQCN Server Details

### 3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The abstract data model for this protocol comprises elements that are private to this protocol and others that are shared between multiple MSMQ protocols that are collocated at a common queue manager. The shared abstract data model is defined in [\[MS-MQDMPR\]](#) section 3.1.1, and the relationship between this protocol, a queue manager, and other protocols that share a common queue manager is described in [\[MS-MQSO\]](#).

Section [3.2.1.1](#) references the elements from the shared data model that are manipulated by this protocol.

Section [3.2.1.2](#) describes the elements that are specific to the server side of this protocol.

#### 3.2.1.1 Shared Data Elements

The server side of this protocol manipulates the shared data elements listed in section [3.1.1.1](#).

#### 3.2.1.2 Local Data Elements

In addition to the shared data elements, the server side of this protocol manipulates the following abstract data model element:

**Change Notification Queue** data element.

##### 3.2.1.2.1 Change Notification Queue

Change Notification Queue is a local private notification queue. The name for this queue is in the format specified by the following ABNF rule.

```
QueuePathName = (Computer "\private$\notify_queue$")
Computer      = 1*256(VCHAR) ; computer where the queue manager resides
VCHAR         = %x21-7E
```

### 3.2.2 Timers

None.

### 3.2.3 Initialization

The implementation of this protocol SHOULD open the local private change notification queue. [<8>](#) This queue MUST be opened by raising an [Open Queue](#) event ([\[MS-MQDMPR\]](#) section 3.1.7.1.5). This event MUST be raised with the following arguments:

- *iFormatName* := NOTIFY\_QUEUE\_PRIVATE as defined in section [1.9](#).
- *iRequiredAccess* := **ReceiveAccess**.
- *iSharedMode* := **DenyNone**.

This event returns *rStatus* and *rOpenQueueDescriptor*. If *rStatus* is MQ\_OK, the value of *rOpenQueueDescriptor* is stored for reuse when dequeuing messages from this queue. Otherwise, if *rStatus* is different than MQ\_OK, this protocol cannot work correctly.

After the change notification queue is open, the implementation MUST start a loop for the lifetime of the server side of this protocol for dequeuing and processing change notification messages:

- Raise the [Dequeue Message](#) event ([\[MS-MQDMPR\]](#) section 3.1.7.1.10). This event MUST be raised with the following arguments:
  - *iQueueDesc* := The OpenQueueDescriptor returned when opening the change notification queue.

This event returns *rMessage*.

### 3.2.4 Higher-Layer Triggered Events

None.

### 3.2.5 Processing Events and Sequencing Rules

The server MUST initialize the local private change notification queue as specified in section [3.2.3](#). When a [Dequeue Message](#) event ([\[MS-MQDMPR\]](#) section 3.1.7.1.10) is raised, the server MUST process the returned *rMessage* as follows:

- Let *NotificationMessage* be initialized to *rMessage.Body* parsed as a [Change Notification Message \(section 2.2.4\)](#).
- Verify the value of field **NotificationMessage.Version**, and:
  - If the value of field **NotificationMessage.Version** is 0x01, field **NotificationMessage.Data** MUST be interpreted as an array of instances of [Notification Update \(section 2.2.6\)](#). The number of instances is determined by the value of field **NotificationMessage.NumberOfUpdateNotifications**. For receiving a change notification message version 0x01, see section [3.2.5.1](#).
  - Otherwise, if the value of field **NotificationMessage.Version** is 0x02, the field **NotificationMessage.Data** MUST be interpreted as an instance of [Notification Body \(section 2.2.5\)](#). For receiving a change notification message version 0x02, see section [3.2.5.2](#).



After a notification message is processed, another Dequeue Message event MUST be generated as specified in section [3.2.3](#). This process is repeated for the lifetime of the queue manager.

### 3.2.5.1 Processing a Change Notification Message version 0x01

For a change notification message version 0x01, **NotificationMessage.Data** is an array of [Notification Updates \(section 2.2.6\)](#). For each Notification Update in the array, the server MUST update the state of the queue or queue manager object referenced by the **PathName** or **GuidIdentifier** field.

To process the update, the server MUST perform the following steps for each Notification Update in the array:

- The queue manager MUST verify that the change notification message *rMessage* was authenticated, according to the following rules:
  - If *rMessage*.**SenderIdIdentifierType** does not equal **SenderIdIdentifierType.QueueManagerIdentifier**, the server MUST disregard the [Change Notification Message \(section 2.2.4\)](#) and perform no further processing.
  - If *rMessage*.**AuthenticationLevel** has a value of 0x0 (**AuthenticationLevel.None**), the server MUST disregard the Change Notification Message and perform no further processing.
- The object type (either a queue object or a queue manager object) in a Notification Update MUST be determined according to the following rules:
  - If the **Command** field is set to 0x00 (created) or 0x01 (updated), the server MUST perform the following actions:
    - If the value of **PropertyId[0]** is greater than 1,000, the object type MUST be set to  $(\text{PropertyId}[0] - 1,000) / 100$ .
    - If the value of **PropertyId[0]** is less than or equal to 1,000, the object type MUST be set to  $\text{PropertyId}[0] / 100$ .
    - If the object type does not equal 1 (a queue object) or 2 (a queue manager object), the server MUST disregard the Change Notification Message and perform no further processing.
- Once the object type is identified, the following actions MUST be performed:
  - If the **Command** field equals 0x00 (created) and object type is 1 (a queue object), the following processing rules MUST be performed:
    - Let *newQueue* be a new **Queue** ([\[MS-MQDMPR\] section 3.1.1.2](#)) ADM element instance.
    - Raise a [Set Queue Defaults](#) ([\[MS-MQDMPR\] section 3.1.7.1.34](#)) event with the following arguments:
      - *iQueue* := *newQueue*
    - Set the following attributes on *newQueue*:

**Type** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_TYPE](#)) ]

**Identifier** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_INSTANCE](#)) ]

**BasePriority** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_BASEPRIORITY](#)) ]  
**Journaling** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_JOURNAL](#)) ]  
**Quota** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_QUOTA](#)) ]  
**JournalQuota** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_JOURNAL\\_QUOTA](#)) ]  
**CreateTime** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_CREATE\\_TIME](#)) ]  
**ModifyTime** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_MODIFY\\_TIME](#)) ]  
**Security** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_SECURITY](#)) ]  
**PathName** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_PATHNAME](#)) ]  
**Label** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_LABEL](#)) ]  
**Authentication** := **PropertyValue** [ IndexOf([PROPID\\_Q\\_AUTHENTICATE](#)) ]  
**PrivacyLevel** := **PropertyValue** [ IndexOf([PROPID\\_Q\\_PRIV\\_LEVEL](#)) ]  
**Transactional** := **PropertyValue** [ IndexOf([PROPID\\_Q\\_TRANSACTION](#)) ]  
**Scope** := **PropertyValue** [ IndexOf([PROPID\\_Q\\_SCOPE](#)) ]

- Raise a [Create Queue](#) ([\[MS-MQDMPR\]](#) section 3.1.7.1.3) event with the following arguments:
  - *iQueue* := *newQueue*.
  - *iSkipDirectory* := True.
- If the **Command** field equals 0x01 (updated) and object type is 1 (a queue object), the following processing rules MUST be performed:
  - Update the corresponding **Queue** ADM element instance in **QueueManager.QueueCollection** with the following values:

**BasePriority** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_BASEPRIORITY](#)) ]  
**Journal** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_JOURNAL](#)) ]  
**Quota** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_QUOTA](#)) ]  
**JournalQuota** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_JOURNAL\\_QUOTA](#)) ]  
**Security** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_SECURITY](#)) ]  
**Authentication** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_AUTHENTICATE](#)) ]  
**PrivacyLevel** := **PropertyValue**[ IndexOf([PROPID\\_Q\\_PRIV\\_LEVEL](#)) ]

All other PROPIDs are not used, and they MUST be ignored.

- If the **Command** field equals 0x01 (updated) and object type is 2 (a queue manager object), the following processing rules MUST be performed:

- Update the attributes of the **QueueManager** ([\[MS-MQDMPR\]](#) section 3.1.1.1) ADM element instance with the following values:

**QueueManagerQuota** := **PropertyValue**[ IndexOf([PROPID\\_QM\\_QUOTA](#)) ]

**JournalQuota** := **PropertyValue**[ IndexOf([PROPID\\_QM\\_JOURNAL\\_QUOTA](#)) ]

**Security** := **PropertyValue**[ IndexOf([PROPID\\_QM\\_SECURITY](#)) ]

All other PROPIDs are not used, and they MUST be ignored.

- If the **Command** field equals 0x02 (deleted), the object type MUST be set to the value of **PropertyValue**[ 1 ]. If the object type does not equal MQDS\_QUEUE, the server MUST disregard the Change Notification Message and perform no further processing; otherwise, the following processing rules MUST be performed:
  - Raise a [Delete Queue](#) ([\[MS-MQDMPR\]](#) section 3.1.7.1.4) event with the following arguments:
    - *iQueue* := A reference to the **Queue** ADM element instance from **QueueManager.QueueCollection** whose identifier is equal to **GuidIdentifier**.
    - *iSkipDirectory* := True.

Section [3.2.1](#) specifies the abstract model that can be used as a reference to maintain data specific to **Queue** and **QueueManager** ADM elements in a queue manager that implements this protocol.

### 3.2.5.2 Processing a Change Notification Message version 0x02

For a change notification message version 0x02, **NotificationMessage.Data** is a [Notification Body](#). The server MUST update the state of the queue or machine object referenced by the GUID contained in **ObjectGuid**. If the value contained in **Event** indicates creation or update, the queue manager MUST request information from the directory service specified within DomainController for the updating process, as specified in [\[MS-MQDMPR\]](#).

The following actions MUST be performed upon the reception of a notification message:

- If **Event** is set to 1 (create queue), the following processing rules MUST be performed:
  - Raise a [Read Directory](#) ([\[MS-MQDMPR\]](#) section 3.1.7.1.20) event with the following arguments:
    - *iDirectoryObjectType* := "Queue".
    - *iFilter* := "Identifier" EQUALS ObjectGuid
    - *iAttributeList* := { Type, Identifier, BasePriority, Journal, Quota, JournalQuota, CreateTime, ModifyTime, Security, Pathname, Label, Authentication, PrivacyLevel, Transactional, MulticastAddress, DirectoryPath }
  - Raise a [Create Queue](#) ([\[MS-MQDMPR\]](#) section 3.1.7.1.3) event with the following arguments:
    - *iQueue* := A copy of the **Queue** ([\[MS-MQDMPR\]](#) section 3.1.1.2) ADM element instance returned by the Read Directory event.
    - *iSkipDirectory* := True.

- If **Event** is set to 2 (change queue), the following processing rules MUST be performed:
  - Raise a Read Directory event with the following arguments:
    - *iDirectoryObjectType* := "Queue".
    - *iFilter* := "Identifier" EQUALS ObjectGuid
    - *iAttributeList* := { Type, Identifier, BasePriority, Journal, Quota, JournalQuota, CreateTime, ModifyTime, Security, Pathname, Label, Authentication, PrivacyLevel, Transactional, MulticastAddress, DirectoryPath }
  - Update the corresponding **Queue** ADM element instance in **QueueManager.QueueCollection** with the **Queue** ADM element instance returned by the Read Directory event.
- If **Event** is set to 3 (delete queue), the following processing rules MUST be performed:
  - Raise a [Delete Queue](#) ([MS-MQDMPR] section 3.1.7.1.4) event with the following arguments:
    - *iQueue* := A reference to the **Queue** ADM element instance in **QueueManager.QueueCollection**, where **Queue.Identifier** is equal to **ObjectGuid**.
    - *iSkipDirectory* := True.
- If **Event** is set to 4 (change queue manager), the following processing rules MUST be performed:
  - Raise a Read Directory event with the following arguments:
    - *iDirectoryObjectType* := "QueueManager".
    - *iFilter* := "Identifier" EQUALS ObjectGuid
    - *iAttributeList* := { QueueManagerQuota, JournalQuota, Security }
  - Update the **QueueManager** ([MS-MQDMPR] section 3.1.1.1) ADM element instance with the **QueueManager** ADM element instance returned by the Read Directory event.

Section [3.2.1](#) describes the abstract model that can be used as a reference to maintain data specific to queue and queue manager data elements in a queue manager that implements this protocol.

### 3.2.6 Timer Events

There are no timer events.

### 3.2.7 Other Local Events

There are no other local events.

## 3.3 MQCN Client Details

### 3.3.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations

adhere to this model as long as their external behavior is consistent with that described in this document.

The abstract data model for this protocol comprises elements that are private to this protocol and others that are shared between multiple MSMQ protocols that are colocated at a common queue manager. The shared abstract data model is defined in [\[MS-MQDMPR\]](#) section 3.1.1, and the relationship between this protocol, a queue manager, and other protocols that share a common queue manager is described in [\[MS-MQSO\]](#).

Section [3.3.1.1](#) references the elements from the shared data model that are manipulated by this protocol.

### 3.3.1.1 Shared Data Elements

The server side of this protocol manipulates the shared data elements listed in section [3.1.1.1](#). In addition, this protocol manipulates the following abstract data model elements from the shared abstract data model defined in [\[MS-MQDMPR\]](#) section 3.1.1.

**OutgoingQueue:** The data element as defined in [\[MS-MQDMPR\]](#) section 3.1.1.3.

### 3.3.2 Timers

None.

### 3.3.3 Initialization

The implementation of the client side of MQCN SHOULD open the **OutgoingQueue** to send change notification messages. This queue MUST be opened by raising an **Open Queue** event. This event MUST be raised with the following arguments:

- `iFormatName` := `OutgoingQueue` with the following attributes values:
  - `DestinationFormatName` := `NOTIFY_QUEUE_PRIVATE` as defined in section [1.9](#).
  - `NextHops` := Empty string.
- `iRequiredAccess` := `SendAccess`.
- `iSharedMode` := `DenyNone`.

This event returns *rStatus* and *rOpenQueueDescriptor*. If *rStatus* is `MQ_OK`, the value of *rOpenQueueDescriptor* is assigned to *openNotifyQueueDescriptor* to be used when enqueueing messages to this queue. Otherwise, if *rStatus* is different than `MQ_OK`, this protocol cannot work correctly.

### 3.3.4 Higher-Layer Triggered Events

The operation of this protocol is initiated and subsequently driven by the following higher-layer triggered event:

- Send Change Notification.

#### 3.3.4.1 Send Change Notification

This event MUST be generated with the following arguments:

- *iOperation*: An enumeration identifying the subject of the notification. Valid values are QueueCreation, QueueUpdate, QueueDeletion, and QueueManagerUpdate.
- *iDirectoryObject*: Reference to a queue or QueueManager object that contains the attributes and values of a queue or QueueManager object.

Return Values:

None.

The following actions **MUST** be performed to process this event:

- Prepare a change notification message of either version 0x01 or 0x02. [<9>](#)
  - For version 0x02, use the rules specified in section [3.3.5.1](#).
  - For version 0x01, use the rules specified in section [3.3.5.2](#).
- Send the prepared change notification message following the rules specified in section [3.3.5.3](#).

### 3.3.5 Processing Events and Sequencing Rules

#### 3.3.5.1 Preparing a Change Notification Message Version 0x02

The queue manager **MUST** create a [Change Notification Message](#). The [Notification Body](#) **MUST** be a part of the Change Notification Message, and the Change Notification Message **MUST** be included within a message. Finally, the message **MUST** be sent through MSMQ to the destination queue manager.

The Notification Body, which is specified in section [2.2.5](#), **MUST** be created as follows: It **MUST** be set to a Unicode string in the format specified by the ABNF rule in section [2.2.5](#). **Event** **MUST** contain 1 to indicate that a queue object was created; or 2 to indicate that a queue object was changed; or 3 to indicate that a queue object was deleted; or 4 to indicate that a machine object was changed. [<10>](#) **ObjectGuid** **MUST** contain the GUID of the queue or the machine object that was changed. **DomainController** **MUST** contain the directory service name where the queue or a machine object was changed.

The Change Notification Message, which is specified in section [2.2.4](#), **MUST** be created as follows: Field **Version** **MUST** be set to 0x02. Field **NumberOfUpdateNotifications** **MUST** be set to 0x01 to indicate that there is only one update notification in the notification message. A queue manager **MUST** send only one notification message at a time. Field **Data** **MUST** be filled with the Notification Body instance that **MUST** be created as described in the previous paragraph.

Finally, the Change Notification Message **MUST** be sent within a message instance through MSMQ to the destination queue manager. The values and rules that **MUST** be used for sending MSMQ messages are described in section [Sending a Change Notification Message](#).

#### 3.3.5.2 Preparing a Change Notification Message Version 0x01

The directory service **MUST** create a [Change Notification Message](#). [Notification Updates](#) **MUST** be part of the Change Notification Message, and the Change Notification Message **MUST** be included within a message. Finally, the message **MUST** be sent through MSMQ to the destination queue manager.

The Notification Update instance, which is specified in section [2.2.6](#), **MUST** be created as follows:

- Field **Command** MUST be set to 0x00 to indicate that a queue object was created in the directory service; or it MUST be set to 0x01 to indicate that a queue or machine object was updated; or it MUST be set to 0x02 to indicate that a queue object was deleted.
- Field **UseGuid** MUST be set to 0x00 if **Command** is set to 0x00 to indicate that **PathName** is being used. For other values of **Command**, **UseGuid** MUST be set to 0x01 to indicate that **GuidIdentifier** is being used.
- Field **PathName** MUST be set to a null-terminated value if **UseGuid** is set to 0x00. It MUST be omitted if **UseGuid** is set to 0x01.
- Field **GuidIdentifier** MUST be set to a valid value if **UseGuid** is set to 0x01. It MUST be set to the GUID of either the queue object or the machine object that was changed.
- Field **GuidMasterId** MUST be set with the GUID of the directory service that originated the change notification message.
- Field **NumberOfProperties** MUST be set with the number of properties to attach to Notification Update.
- **PropertyId[]** MUST be filled with property identifiers, and its number of elements MUST be equal to **NumberOfProperties**. The set of property identifiers varies as follows.
  - If **Command** is set to 0x00 (created), and the created object is a queue object, the following property ids MUST be included within **PropertyId[]** in any order:

[PROPID\\_Q\\_TYPE](#)

[PROPID\\_Q\\_INSTANCE](#)

[PROPID\\_Q\\_BASEPRIORITY](#)

[PROPID\\_Q\\_JOURNAL](#)

[PROPID\\_Q\\_QUOTA](#)

[PROPID\\_Q\\_JOURNAL\\_QUOTA](#)

[PROPID\\_Q\\_CREATE\\_TIME](#)

[PROPID\\_Q\\_MODIFY\\_TIME](#)

[PROPID\\_Q\\_SCOPE](#)

[PROPID\\_Q\\_SECURITY](#)

[PROPID\\_Q\\_PATHNAME](#)

[PROPID\\_Q\\_LABEL](#)

[PROPID\\_Q\\_AUTHENTICATE](#)

[PROPID\\_Q\\_PRIV\\_LEVEL](#)

[PROPID\\_Q\\_TRANSACTION](#)

This property MUST NOT be included. The receiving queue manager MUST ignore this property if received:

[PROPID\\_Q\\_ADS\\_PATH](#)

- If **Command** is set to 0x01 (updated), and the modified object is a queue object, the following property MUST be included within **PropertyId[]**:

[PROPID\\_Q\\_QMID](#)

The rest of **PropertyId[]** MUST be filled with the identifiers of the properties that were modified for the queue object specified in **GuidIdentifier**.

These properties may be modified:

[PROPID\\_Q\\_JOURNAL](#)

[PROPID\\_Q\\_QUOTA](#)

[PROPID\\_Q\\_BASEPRIORITY](#)

[PROPID\\_Q\\_JOURNAL\\_QUOTA](#)

[PROPID\\_Q\\_SECURITY](#)

[PROPID\\_Q\\_AUTHENTICATE](#)

[PROPID\\_Q\\_PRIV\\_LEVEL](#)

- If **Command** is set to 0x01 (updated), and the modified object is a machine object, the following properties MUST be included within **PropertyId[]** in any order:

[PROPID\\_QM\\_MACHINE\\_ID](#)

[PROPID\\_QM\\_FOREIGN](#)

The rest of **PropertyId[]** MUST be filled with the identifiers of the properties that were modified for the machine object specified in **GuidIdentifier**.

These properties may be modified:

[PROPID\\_QM\\_QUOTA](#)

[PROPID\\_QM\\_JOURNAL\\_QUOTA](#)

[PROPID\\_QM\\_SECURITY](#)

- The following properties may be included but will be ignored on the Server side of this protocol. [<11>](#)

[PROPID\\_QM\\_ADDRESS](#)

[PROPID\\_QM\\_CNS](#)



[PROPID\\_QM\\_MACHINE\\_TYPE](#)

[PROPID\\_QM\\_OS](#)

- If **Command** is set to 0x02 (deleted), and the modified object is a queue object, the **PropertyId[]** MUST be filled as follows:

PropertyId[0] = [PROPID\\_D\\_SCOPE](#)

PropertyId[1] = [PROPID\\_D\\_OBJTYPE](#)

Message Queuing (MSMQ): Data Structures, as specified in [\[MS-MQMQ\]](#), contains further information on the property identifiers used in this section.

- **PropertyValue[]** MUST be filled with the values corresponding to each element of the array of property identifiers contained in **PropertyId[]**. The rules that MUST be followed to set the values for these properties are defined within this document and Message Queuing (MSMQ): Data Structures, as specified in [\[MS-MQMQ\]](#). In addition, some properties MUST be set according to the following rules:
  - If **Command** is set to 0x02 (deleted), and the modified object is a queue object, the **PropertyValue[]** MUST be filled as follows:

PropertyValue[0] = MQDS\_ENTERPRISESCOPE

PropertyValue[1] = MQDS\_QUEUE

The Change Notification Message, which is described in section [2.2.4](#), MUST be created as follows. Field **Version** MUST be set to 0x01. Field **NumberOfUpdateNotifications** MUST be set to the number of Notification Update instances to be included in field **Data**. A directory service can send more than one notification message at a time. Field **Data** MUST be filled with the Notification Update instances that MUST be created as described in the previous paragraph.

Finally, the Change Notification Message MUST be sent within a message instance through MSMQ to the destination queue manager. The values and rules that MUST be used for sending MSMQ messages are specified in section [Sending a Change Notification Message](#).

### 3.3.5.3 Sending a Change Notification Message

The [Change Notification Message](#) MUST be sent within a **Message** data element through MSMQ to the destination queue manager. This section specifies the values that MUST be set that are specific to notification messages.

The sender MUST perform the following actions to process this event:

- Create a **Message** data element with the following attributes:
  - **TimeToReachQueue** := 5 minutes
  - **DeliveryGuarantee** := Express
  - **Class** := Normal
  - **HashAlgorithm** MUST be set. [<12>](#)

- **AuthenticationLevel** := **Sig20**[<13>](#)
- The sender MUST create a buffer of bytes of the size of the Change Notification Message plus 1. This additional byte is unused and MUST not affect the behavior of this protocol. The Change Notification Message MUST be included at the beginning of the created buffer. This buffer MUST be assigned to **Message.Body**.
- If the message is version 0x01, the sender MUST perform the following actions:
  - Set **Message.SenderIdentifier** to **QueueManager.Identifier**.
  - Set **Message.SenderIdentifierType** to **QueueManagerIdentifier**.
- Once the message is prepared, the implementation of this protocol MUST generate an Enqueue Message event. This event MUST be raised with the following arguments:
  - *iQueue* := The destination queue opened during the initialization phase in section [3.3.3](#), (openNotifyQueueDescriptor).
  - *iMessage* := The message prepared in this section.

### 3.3.6 Timer Events

There are no timer events.

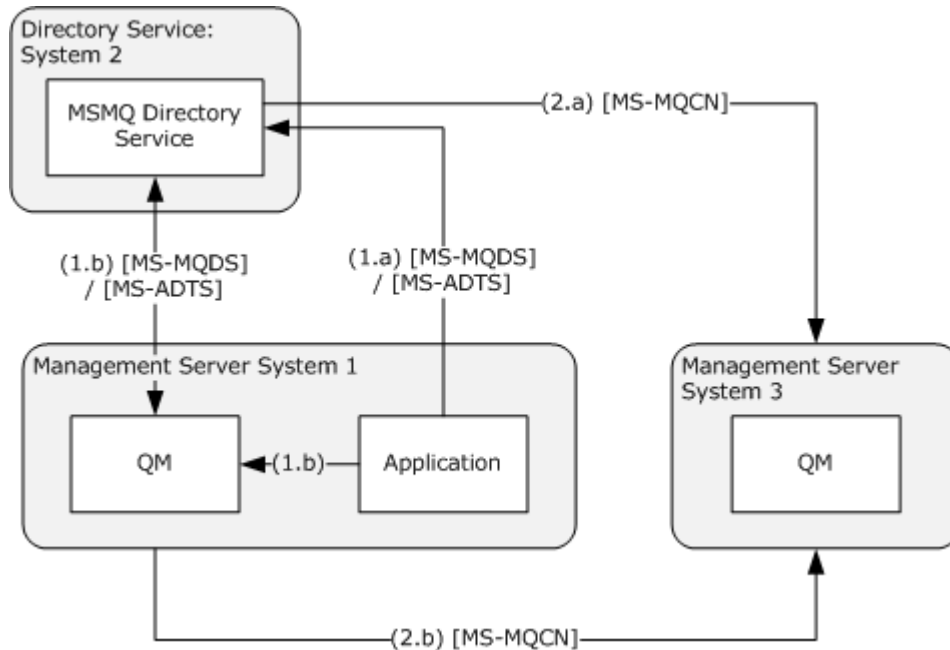
### 3.3.7 Other Local Events

There are no other local events.

## 4 Protocol Examples

### 4.1 Management Client Update Profile

This example describes how an application can create a **public queue** on a remote queue manager. The application profile consists of MSMQ Directory Service and two **MSMQ management servers**. Both MSMQ management servers are aware of the MSMQ Directory Service and can access it. The following steps show how an application can create a public queue on a remote machine.



**Figure 1: Creation of a public queue on a remote queue manager**

1. An application on System 1 requests that a public queue be created on System 3. Microsoft Message Queuing (MSMQ) first notifies the MSMQ Directory Service and updates it by creating a queue object. This update can happen in two ways:
  - This request can be made directly through the MSMQ runtime. This communication is facilitated over the Message Queuing (MSMQ): Directory Service Protocol (as specified in [\[MS-MQDS\]](#)) for MSMQ Version 1 and is facilitated over **Active Directory** (as specified in [\[MS-ADTS\]](#)) for MSMQ Version 2, onward.
  - If the runtime is unable to create the object, the request is handed to the local queue manager. In this case, the queue manager on System 1 updates the MSMQ Directory Service using the Message Queuing (MSMQ): Directory Service Protocol (as specified in [\[MS-MQDS\]](#)) or Active Directory (as specified in [\[MS-ADTS\]](#)).
2. The actual creation of the public queue is handled via the queue manager on System 3. The queue manager 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.

- If MSMQ Version 1.0 directory service is running, the MSMQ Directory Service informs the queue manager via the MSMQ: Directory Service Change Notification Protocol, as shown in arrow 2.a.
- If Active Directory is used, the queue manager on System 1 informs the queue manager on System 3 via the MSMQ: Directory Service Change Notification Protocol, as shown in arrow 2.b.

## 5 Security

### 5.1 Security Considerations for Implementers

This protocol MAY allow denial-of-service (DoS) attacks to the server. An implementation of this protocol SHOULD develop mechanisms to prevent such attacks. One of these mechanisms MAY be to process notifications only every few minutes. [<14>](#)

### 5.2 Index of Security Parameters

There are no security parameters for this protocol.

## 6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows NT® operating system
- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

[<1> Section 1.6:](#) On Windows NT and Windows 2000, the MSMQ Directory Service notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. Otherwise, on Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. If an **MSMQ routing server** is enabled, the path that the message takes between the local queue manager and the destination queue manager can be altered, as indicated in [\[MS-MQBR\]](#).

[<2> Section 1.6:](#) On Windows NT and Windows 2000, the MSMQ Directory Service notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. Otherwise, on Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. If an MSMQ routing server is enabled, the path that the message takes between the local queue manager and the destination queue manager can be altered, as indicated in [\[MS-MQBR\]](#).

[<3> Section 1.6:](#) On Windows NT and Windows 2000, queue managers process notification messages as soon as they arrive. In contrast, on Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, notification messages are delayed by up to 15 minutes to prevent denial-of-service attacks. If the last change notification message was processed more than 14 minutes ago, the queue manager sets a timer for one minute and waits for additional change notification messages to arrive. If the last change notification message was processed 14 or less minutes ago, the timer is set for the remainder of the 15 minutes instead of one minute. When the timer expires, the queue manager processes all received change notification messages. If 100 or more change notification messages arrive while the

queue manager is waiting for the timer to expire, it ignores the messages and instead updates the state of all the public queue objects that it owns with the state of those objects that is stored in the directory service (as described in section [3.2.5.2](#) for a change queue event) when the timer expires.

[<4> Section 1.7:](#) On Windows NT and Windows 2000, queue managers process messages only from the directory service. Other messages are ignored. On Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, queue managers process messages either from the directory service or from queue managers. Other messages are ignored.

[<5> Section 2.2.4:](#) On Windows NT and Windows 2000, the MSMQ Directory Service notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. Otherwise, on Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. If an MSMQ routing server is enabled, the path that the message takes between the local queue manager and the destination queue manager can be altered, as indicated in [\[MS-MQBR\]](#).

[<6> Section 2.3:](#) For Windows NT and Windows 2000, this protocol uses the Message Queuing (MSMQ): Directory Service Protocol [\[MS-MQDS\]](#).

[<7> Section 2.3:](#) For the Message Queuing (MSMQ): Directory Service Protocol [\[MS-MQDS\]](#), the Directory Service schema elements are described in [\[MS-MQDS\]](#) sections [2.2.10](#) and [3.1.4.21.1](#) through [3.1.4.21.4](#).

[<8> Section 3.2.3:](#) On Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, when a queue manager is running in **workgroup mode**, the notification queue is not created, and the protocol is not initialized.

[<9> Section 3.3.4.1:](#) On Windows NT and Windows 2000, the MSMQ Directory Service notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. Otherwise, on Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol. If an MSMQ routing server is enabled, the path that the message takes between the local queue manager and the destination queue manager can be altered, as indicated in [\[MS-MQBR\]](#).

[<10> Section 3.3.5.1:](#) On all Windows versions, if a message is received and the value of *Event* is not 1, 2, 3, or 4, then the message is ignored.

[<11> Section 3.3.5.2:](#) For [Change Notification Message](#) Version 0x01, the Windows implementation of the Client side of MQCN will send additional properties which are ignored by the Server side of MQCN. This is done because the Windows implementation takes the data structure it uses in the directory service and passes it as is without removing the additional or unnecessary properties.

[<12> Section 3.3.5.3:](#) MSMQ versions 1 and 2 use MD5, as specified in [\[RFC1321\]](#); versions 2 and 4 use SHA1 as specified in [\[RFC3110\]](#).

[<13> Section 3.3.5.3:](#) MSMQ version 1 sets this value to **Sig10**.

[<14> Section 5.1:](#) On Windows NT and Windows 2000, queue managers process notification messages as soon as they arrive. In contrast, on Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, notification messages are delayed by up to 15 minutes to prevent denial-of-service attacks. If the last change notification message was processed more than 14 minutes ago, the queue manager sets a timer for 1 minute and waits for additional change notification messages to arrive. When the timer expires, the queue manager processes all received change notification messages and clears the timer. If 100

or more change notification messages arrive while the queue manager is waiting for the timer to expire, it ignores the messages and instead refreshes all public queues when the timer expires. If the last change notification message was processed 14 or fewer minutes ago, the timer is set for the remainder of the 15 minutes instead of 1 minute.



## 7 Change Tracking

This section identifies changes that were made to the [MS-MQCN] protocol document between the May 2011 and June 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact [protocol@microsoft.com](mailto:protocol@microsoft.com).

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
<a href="#">1.2 References</a>	Added explanatory statement regarding the removal of the publishing year from Microsoft Open Specification document references.	N	Content updated.
<a href="#">2.2.3.10 VT_LPWSTR</a>	64769 Changed the required initialization value for the NullTerminator field from 0x00 to 0x0000.	N	Content updated.

## 8 Index

### A

Abstract data model  
  client ([section 3.1.1](#) 21, [section 3.3.1](#) 28)  
  server  
    [local data elements](#) 23  
    overview ([section 3.1.1](#) 21, [section 3.2.1](#) 23)  
    [shared data elements](#) 23  
[Applicability](#) 9

### C

[Capability negotiation](#) 9  
[Change tracking](#) 41  
Client  
  abstract data model ([section 3.1.1](#) 21, [section 3.3.1](#) 28)  
  [higher-layer triggered events](#) 29  
  [initialization](#) 29  
  [local events](#) 34  
  message processing ([section 3.1.5](#) 23, [section 3.3.5](#) 30)  
  sequencing rules ([section 3.1.5](#) 23, [section 3.3.5](#) 30)  
  [timer events](#) 34  
  [timers](#) 29

### D

[Data elements mapping](#) 21  
Data model - abstract  
  client ([section 3.1.1](#) 21, [section 3.3.1](#) 28)  
  server  
    [local data elements](#) 23  
    overview ([section 3.1.1](#) 21, [section 3.2.1](#) 23)  
    [shared data elements](#) 23

### E

[Example - management client update profile](#) 35

### F

[Fields - vendor-extensible](#) 9

### G

[Glossary](#) 6  
[GUID](#) 10

### H

Higher-layer triggered events  
  [client](#) 29  
  [server](#) 24

### I

[Implementer - security considerations](#) 37  
[Index of security parameters](#) 37  
[Informative references](#) 7  
Initialization  
  [client](#) 29  
  [server](#) 24  
[Introduction](#) 6

### L

[Local data elements - server](#) 23  
Local events  
  [client](#) 34  
  [server](#) 28

### M

[Management client update profile example](#) 35  
Mapping  
  [data elements](#) 21  
  [queue attributes](#) 22  
  [queue manager attributes](#) 21  
Message processing  
  client ([section 3.1.5](#) 23, [section 3.3.5](#) 30)  
  server ([section 3.1.5](#) 23, [section 3.2.5](#) 24)  
Messages  
  syntax  
    [GUID](#) 10  
    [notification body](#) 17  
    [overview](#) 10  
    [PROPVALUE](#) 12  
    [transport](#) 10

### N

[Normative references](#) 7  
[Notification body](#) 17  
[Notification Header packet](#) 16  
[Notification Update packet](#) 18

### O

[Overview \(synopsis\)](#) 8

### P

[Parameters - security index](#) 37  
[Preconditions](#) 8  
[Prerequisites](#) 8  
[Product behavior](#) 38  
[PROPID constants](#) 10  
[PROPID packet](#) 10  
[PROPVALUE](#) 12

### Q

[Queue attributes mapping](#) 22  
[Queue manager attributes mapping](#) 21

## R

### References

[informative](#) 7

[normative](#) 7

[Relationship to other protocols](#) 8

## S

### Security

[implementer considerations](#) 37

[parameter index](#) 37

### Sequencing rules

client ([section 3.1.5](#) 23, [section 3.3.5](#) 30)

server ([section 3.1.5](#) 23, [section 3.2.5](#) 24)

### Server

#### abstract data model

[local data elements](#) 23

overview ([section 3.1.1](#) 21, [section 3.2.1](#) 23)

[shared data elements](#) 23

[higher-layer triggered events](#) 24

[initialization](#) 24

[local events](#) 28

message processing ([section 3.1.5](#) 23, [section 3.2.5](#) 24)

sequencing rules ([section 3.1.5](#) 23, [section 3.2.5](#) 24)

[timer events](#) 28

[timers](#) 24

### Shared data elements

[client](#) 29

[server](#) 23

[Standards assignments](#) 9

### Syntax

[GUID](#) 10

[notification body](#) 17

[overview](#) 10

[PROPValue](#) 12

## T

### Timer events

[client](#) 34

[server](#) 28

### Timers

[client](#) 29

[server](#) 24

[Tracking changes](#) 41

[Transport](#) 10

### Triggered events - higher-layer

[client](#) 29

[server](#) 24

## V

[Vendor-extensible fields](#) 9

[Versioning](#) 9

[VT\\_BLOB packet](#) 14

[VT\\_BOOL packet](#) 12

[VT\\_CLSID packet](#) 15

[VT\\_CLSID VT\\_VECTOR packet](#) 15

[VT\\_I1 packet](#) 12

[VT\\_I2 packet](#) 12

[VT\\_I4 packet](#) 12

[VT\\_I8 packet](#) 13

[VT\\_LPWSTR packet](#) 14

[VT\\_LPWSTR VT\\_VECTOR packet](#) 15

[VT\\_UI1 packet](#) 13

[VT\\_UI2 packet](#) 13

[VT\\_UI4 packet](#) 13

[VT\\_UI4 VT\\_VECTOR packet](#) 15

[VT\\_UI8 packet](#) 14