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

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.

Date	Revision History	Revision Class	Comments
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.

Table of Contents

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 Microsoft Message Queuing (MSMQ) **messages**.

1.1 Glossary

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

Active Directory (AD)
Domain
Domain Controller (DC)
Globally Unique Identifier (GUID)
Little-Endian
Unicode

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

Workgroup mode

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

Connected Network
Dependent Client
Enterprise
Independent Client
Message
Message Body
Message Property
Message Queue
Microsoft Message Queuing (MSMQ)
MSMQ Directory Service
MSMQ Queue Manager
MSMQ Site
MSMQ Site Link
Private Queue
Public Queue
Queue
Queue Manager
System Queue

The following terms are specific to this document:

Directory Service: An entity that maintains a collection of objects. These objects can be remotely manipulated either by the Message Queuing (MSMQ): Directory Service Protocol, as

specified in [MS-MQDS], or by the Lightweight Directory Access Protocol (v3), as specified in [RFC2251].

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

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. 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", June 2007.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[MS-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

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

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

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

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

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

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", January 2007.

[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.ietf.org/rfc/rfc2119.txt

[RFC2251] Wahl, M., Howes, T., and Kille, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997, http://www.ietf.org/rfc/rfc2251.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

[RFC4234] Crocker, D., Ed. and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, http://www.ietf.org/rfc/rfc4234.txt

[UNICODE] The Unicode Consortium, "Unicode Home Page", 2006, http://www.unicode.org/

1.2.2 Informative References

[MSDN-MQEIC] Microsoft Corporation, "Message Queuing Error and Information Codes", http://msdn2.microsoft.com/en-gb/library/ms700106.aspx

1.3 Protocol Overview (Synopsis)

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 a queue and/or consume application messages from a queue. 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**, **site 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 Message Queuing (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 objects owned by the queue manager being notified. 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 Microsoft Message Queuing (MSMQ) as its transport protocol.

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> Applicable scenarios include cases in which a queue manager that is an owner of a queue or of machine objects 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 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 does 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_NAME	<Machine Name>\private$\notify_queue$	None.
NOTIFY_QUEUE_ID	3	None.

2 Messages

The following sections specify how MSMQ: Directory Service Change Notification Protocol messages are transported and MSMQ: Directory Service Change Notification Protocol message syntax.

2.1 Transport

Notifications are sent within messages that are transported as MSMQ messages. The server side of this protocol MUST maintain a notification queue to receive these messages. The client side of this protocol MUST send notifications within MSMQ messages to the server notification queue.

2.2 Message Syntax

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

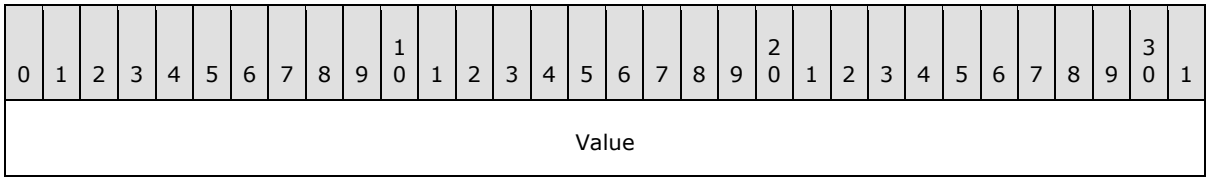
Message or type	Description
GUID	**Globally unique identifier**
PROPID	Property identifier
PROPVALUE	Property value
PROPID_D_SCOPE	Scope of a deletion notification
PROPID_D_OBJTYPE	Type of an object
Notification Header	Header of a notification
Notification Body	Body of a notification. Used by a queue manager.
Notification Update	Alternative body notification. Used by a directory service.

2.2.1 GUID

This specification uses a globally unique identifier. 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.



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].

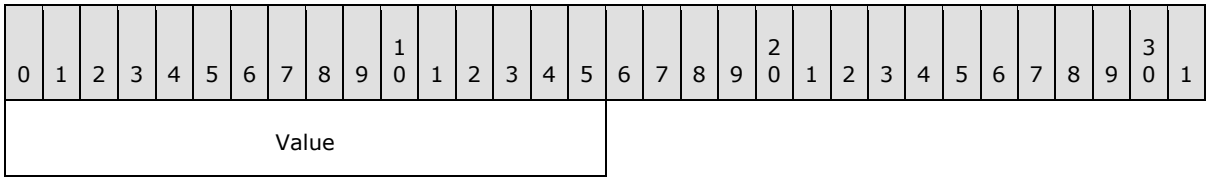
PROPID Constant	Value
PROPID_Q_ADS_PATH	126
PROPID_Q_AUTHENTICATE	111
PROPID_Q_BASEPRIORITY	106
PROPID_Q_CREATE_TIME	109
PROPID_Q_INSTANCE	101
PROPID_Q_JOURNAL	104
PROPID_Q_JOURNAL_QUOTA	107
PROPID_Q_LABEL	108
PROPID_Q_MODIFY_TIME	110
PROPID_Q_MULTICAST_ADDRESS	125
PROPID_Q_PATHNAME	103
PROPID_Q_PRIV_LEVEL	112
PROPID_Q_QMID	115
PROPID_Q_QUOTA	105
PROPID_Q_SECURITY	1101
PROPID_Q_TRANSACTION	113
PROPID_Q_TYPE	102
PROPID_QM_FOREIGN	219
PROPID_QM_JOURNAL_QUOTA	215
PROPID_QM_MACHINE_ID	202
PROPID_QM_QUOTA	214
PROPID_QM_SECURITY	1201

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.

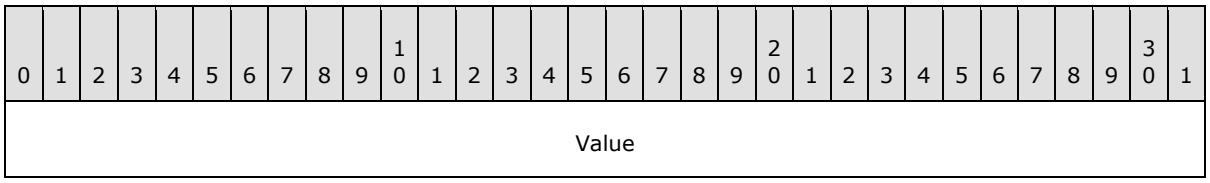
2.2.3.1 VT_I2

The value of a VT_I2-type property MUST be a 2-byte signed integer. It MUST be formatted in little-endian byte order.



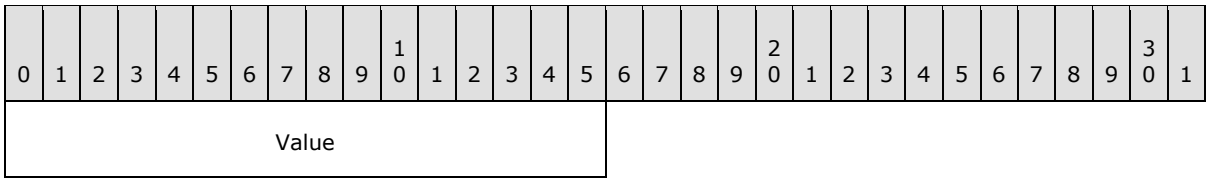
2.2.3.2 VT_I4

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.

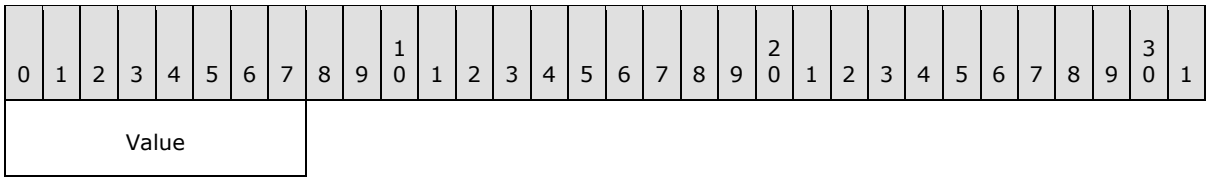


Value (2 bytes): The values MUST be defined as follows.

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

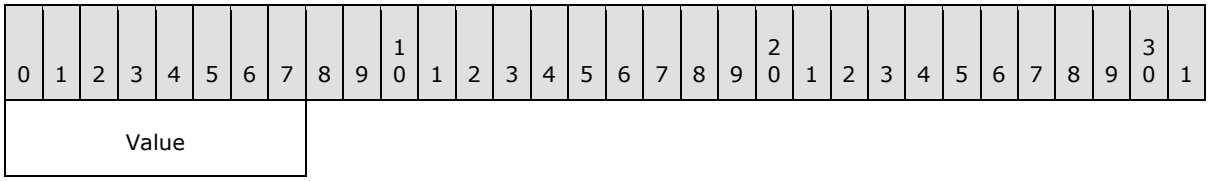
2.2.3.4 VT_I1

The value of a VT_I1-type property MUST be a 1-byte signed integer.



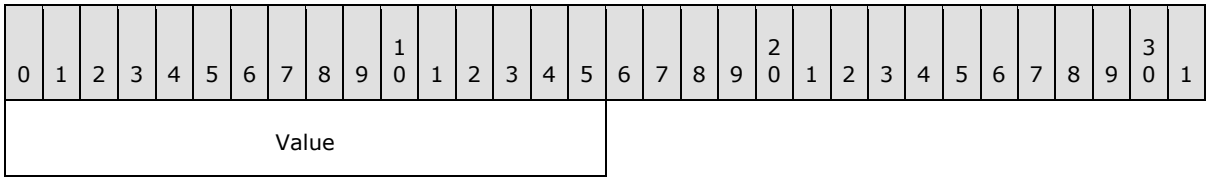
2.2.3.5 VT_UI1

The value of a VT_UI1-type property MUST be a 1-byte unsigned integer.



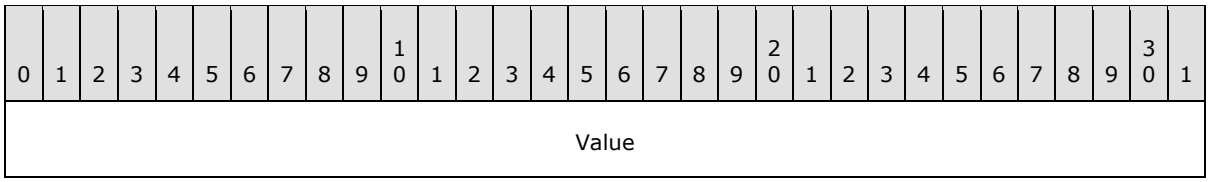
2.2.3.6 VT_UI2

The value of a VT_UI2-type property MUST be a 2-byte unsigned integer. It MUST be formatted in little-endian byte order.



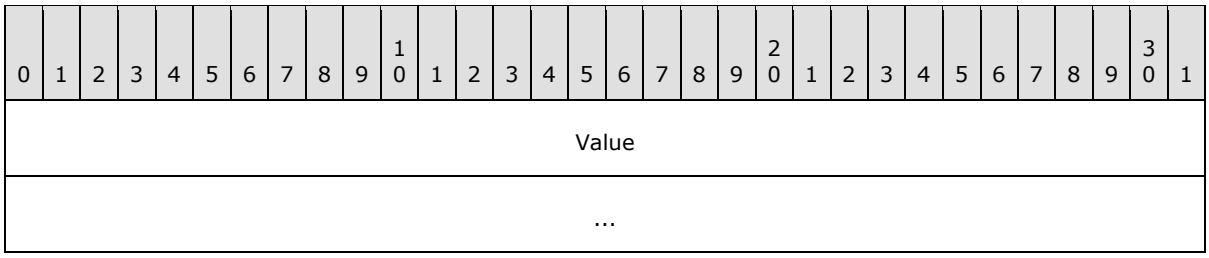
2.2.3.7 VT_UI4

The value of a VT_UI4-type property MUST be a 4-byte unsigned integer. It MUST be formatted in little-endian byte order.



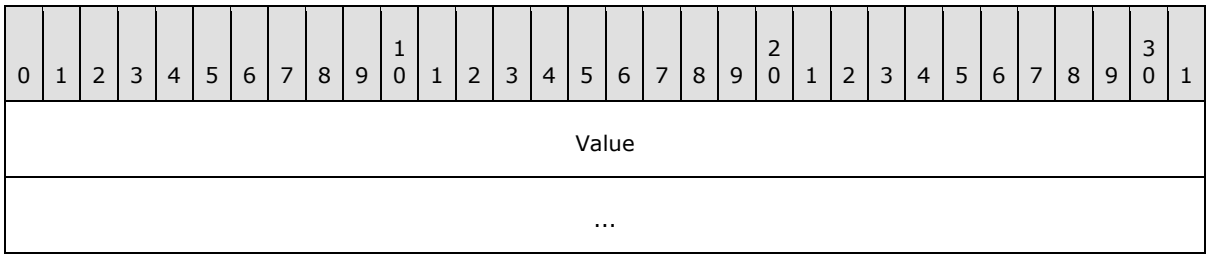
2.2.3.8 VT_I8

The value of a VT_I8-type property MUST be an 8-byte signed integer. It MUST be formatted in little-endian byte order.



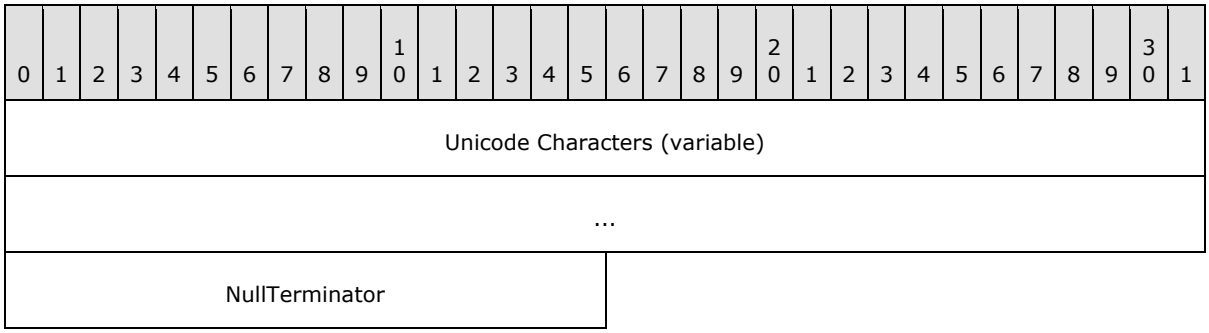
2.2.3.9 VT_UI8

The value of a VT_UI8-type property MUST be an 8-byte unsigned integer. It MUST be formatted in little-endian byte order.



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.

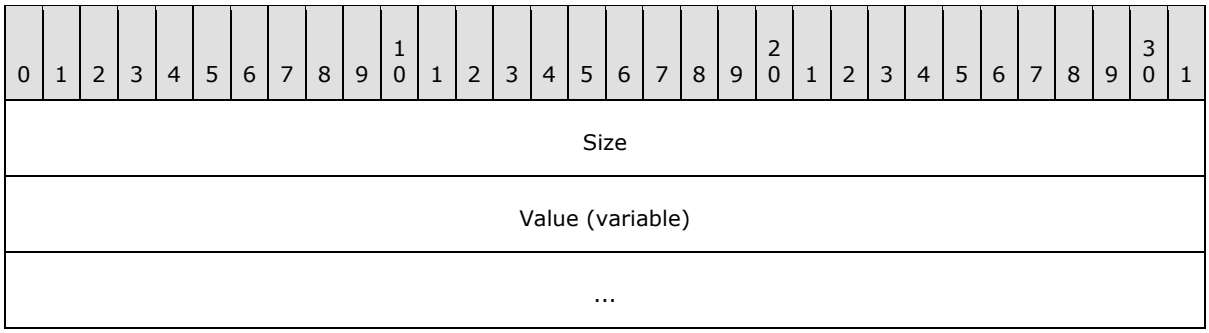


Unicode Characters (variable): Contains a variable-length array of Unicode characters. Each array element MUST be set to a 16-bit Unicode character.

NullTerminator (2 bytes): Contains a Unicode character. This field MUST be set to a value of 0x00.

2.2.3.11 VT_BLOB

The value of a VT_BLOB property MUST be a counted array of unsigned bytes.



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.

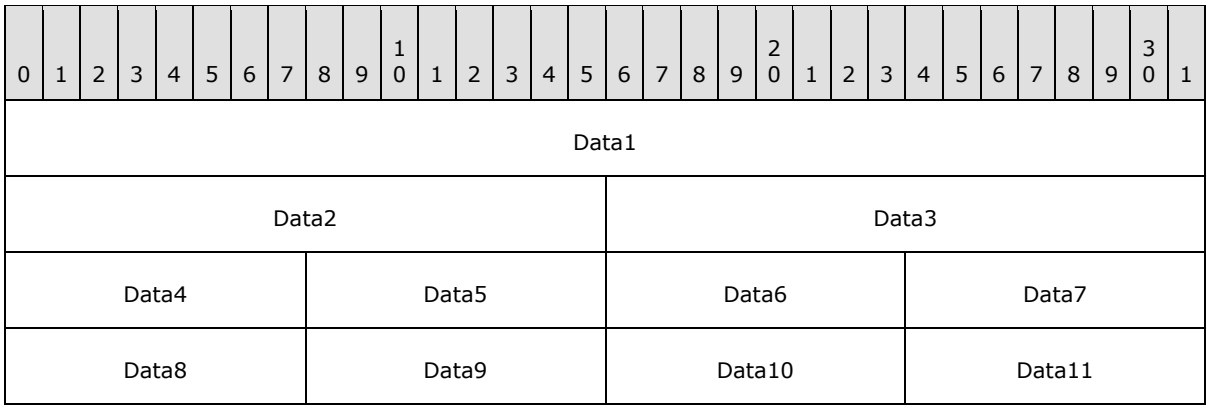
2.2.3.12 VT_CLSID

The value of a VT_CLSID-type property MUST be a **GUID** value (as specified in [MS-DTYP] section **2.3.2**).

Value

Type: **GUID**.

A **GUID** is a 16-byte structure intended to serve as a unique identifier for an object.



Data1 (4 bytes): A 32-bit **LONG** integer.

Data2 (2 bytes): A 16-bit **SHORT** integer.

Data3 (2 bytes): A 16-bit **SHORT** integer.

Data4 (1 byte): A 1-byte **CHAR** value.

Data5 (1 byte): A 1-byte **CHAR** value.

Data6 (1 byte): A 1-byte **CHAR** value.

Data7 (1 byte): A 1-byte **CHAR** value.

Data8 (1 byte): A 1-byte **CHAR** value.

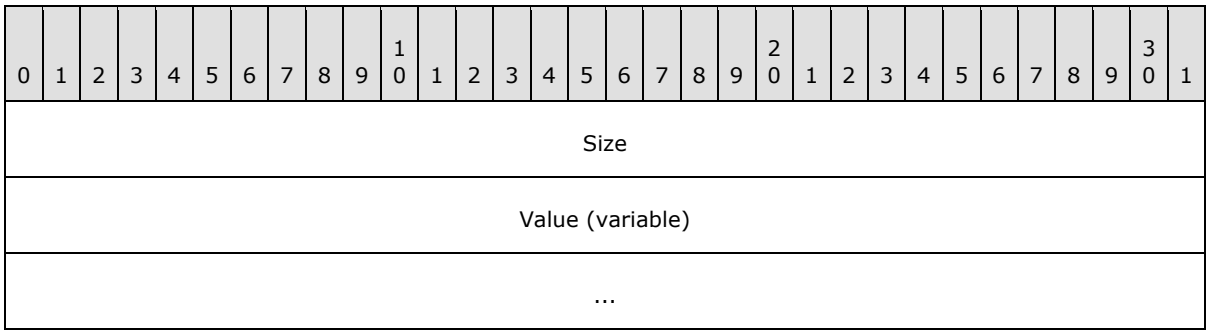
Data9 (1 byte): A 1-byte **CHAR** value.

Data10 (1 byte): A 1-byte **CHAR** value.

Data11 (1 byte): A 1-byte **CHAR** 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.

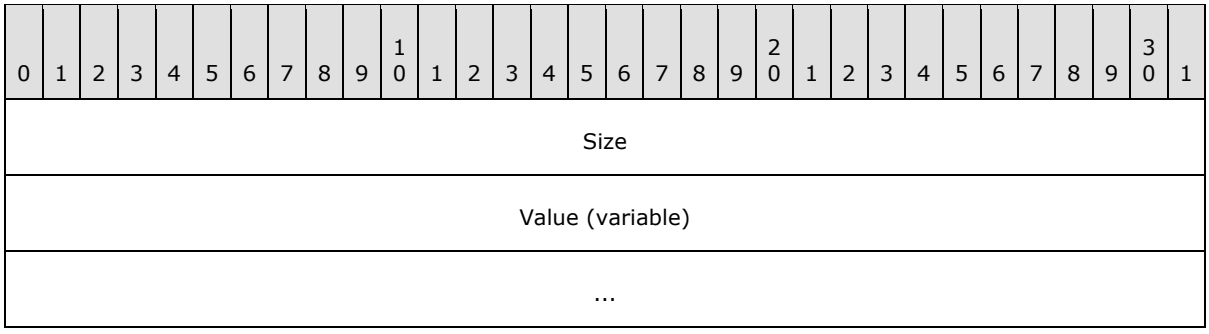


Size (4 bytes): A **ULONG** that specifies the size of the array or vector of values. It MUST be formatted in little-endian byte order.

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.

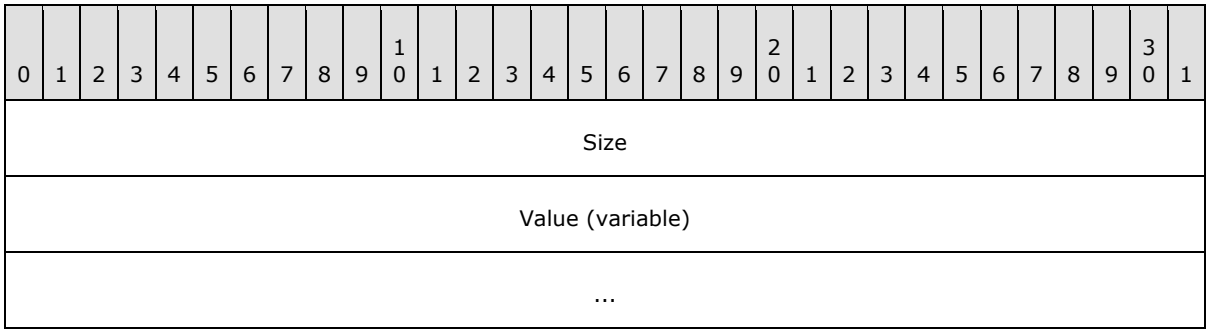


Size (4 bytes): A **ULONG** that specifies the size of the array or vector of values; MUST be formatted in little-endian byte order.

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.



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 PROPID_Q_SYSTEMQUEUE

Value: 5001.

Variant type: VT_UI1.

Description: A value that specifies if a queue is a **system queue**. MUST be one of the following.

Value	Constant	Description
0	NON_SYSTEMQUEUE	Indicates a non-system queue.
1	SYSTEMQUEUE	Indicates a system queue.

2.2.5 PROPID_Q_TIMESTAMP

Value: 5000.

Variant type: VT_BLOB.

Description: A value that specifies the time stamp of a queue.

Value: The value MUST be the number of seconds elapsed since midnight (00:00:00), January 1, 1970 (Coordinated Universal Time). The format of the BLOB is a 4-byte integer representing this number.

2.2.6 PROPID_D_SCOPE

Value: 1403.

Variant type: VT_UI1.

Description: A value that specifies the scope of a deletion notification. MUST be one of the following.

Value	Constant	Description
0	MQDS_SITESCOPE	Indicates a site scope.
1	MQDS_ENTERPRISESCOPE	Indicates an enterprise scope.

2.2.7 PROPID_D_OBJTYPE

Value: 1404.

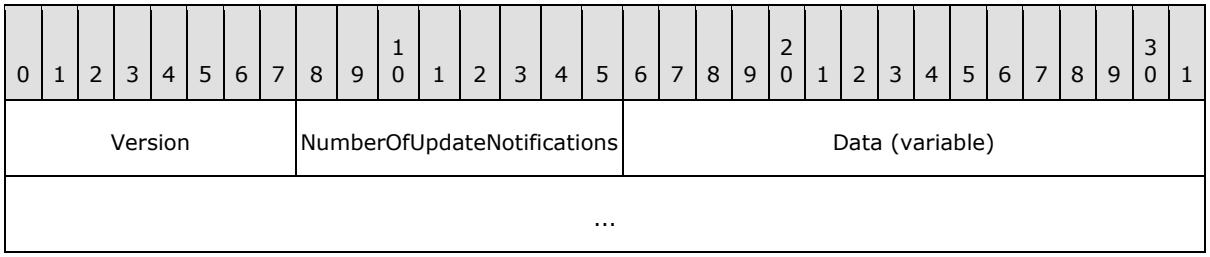
Variant type: VT_UI1.

Description: A value that specifies the type of an object. MUST be one of the following.

Value	Constant	Description
1	MQDS_QUEUE	Object represents a **message queue**.
2	MQDS_MACHINE	Object represents a queue manager.
3	MQDS_SITE	Object represents a site.
4	MQDS_DELETEDOBJECT	Object has been deleted.
5	MQDS_CN	Object represents a connected network.
6	MQDS_ENTERPRISE	Object represents an enterprise.
7	MQDS_USER	Object represents a user.
8	MQDS_SITELINK	Object represents a site link.

2.2.8 Notification Header

A Notification Header 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.



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><6>

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

Value	Meaning
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 that contains a single Notification Body (see section 2.2.9) if *Version* is set to 0x01 or a sequence of Notification Updates (section 2.2.10) if *Version* is set to 0x02. There is no padding between Notification Updates.

2.2.9 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 **domain controller**. 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. This is so that queue managers do not have to be trusted.

<Notification>

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

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

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

</Notification>

Event

Specifies the notification event for a queue or a machine object on which changes were made within a domain controller. This MUST be set to 0, 1, 2, 3, or 4. A value of 0 indicates no event; 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.<7>

ObjectGuid

Specifies the object that was modified within the domain controller. This MUST be set to the **GUID** of the modified object that MUST be either a queue or a machine object.

DomainController

Specifies the domain controller in which the queue or the machine object was changed.

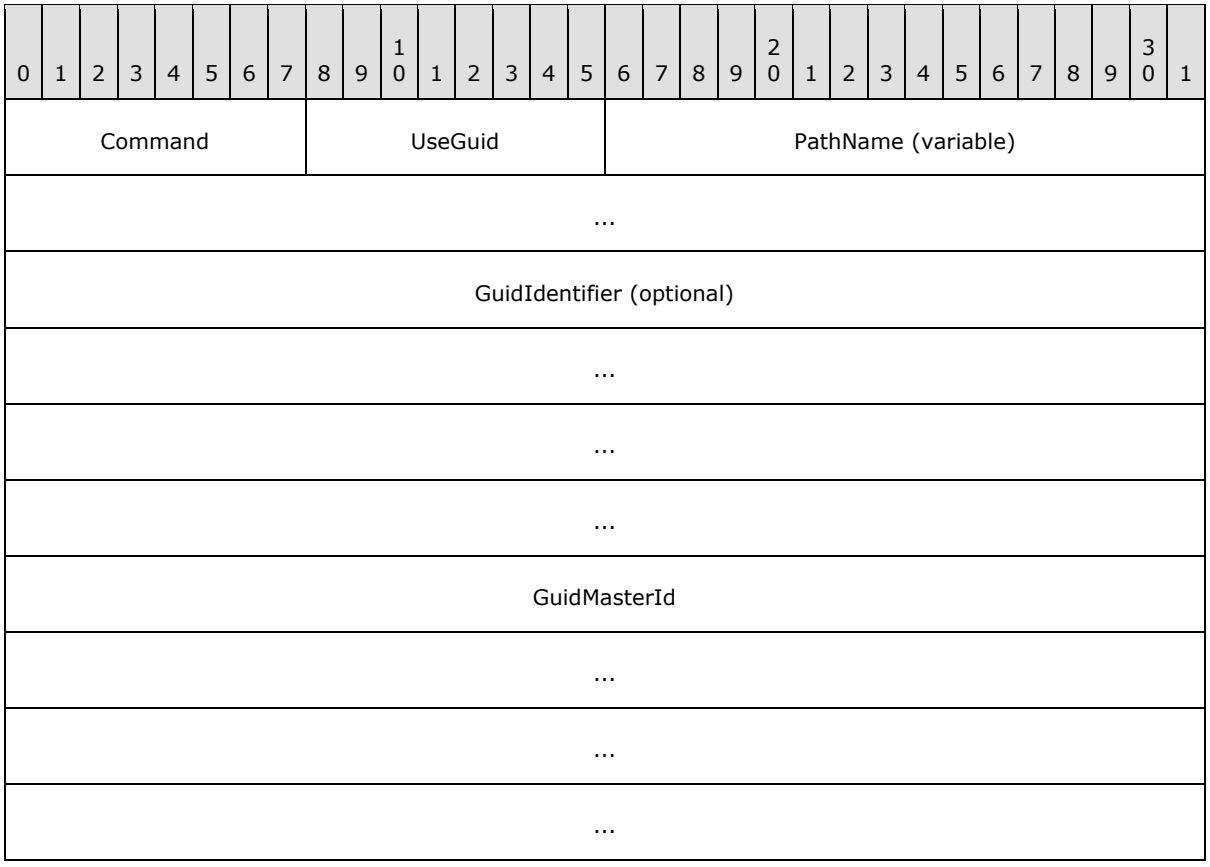
Notification body MUST be set to a Unicode string in the format specified by the following ABNF rule.

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

```
                    "domain-controller"
                   </DomainController>" CRLF
                 = "</Notification>" CRLF

event            = 0|1|2|3|4; where 0: no event, 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

domain-controller = 1*256(VCHAR)
                    ; A directory service name.
```

2.2.10 Notification Update

Notification Update is used to encapsulate a change notification that indicates a change either on a queue or a machine object. 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.



PreviousSequenceNumber	
...	
SequenceNumber	
...	
PurgeSequenceNumber	
...	
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	*PathName* is being used.
0x01	*GuidIdentifier* 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.

PreviousSequenceNumber (8 bytes): An 8-byte array of **UCHAR** that contains the sequence number of the previously sent Notification Update.<8>

SequenceNumber (8 bytes): An 8-byte array of **UCHAR** that contains the sequence number of the current Notification Update.<9>

PurgeSequenceNumber (8 bytes): An 8-byte array of **UCHAR** that contains the sequence number of the Notification Update to be purged.<10>

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 2.2.4 onward define the types for each property ID.

3 Protocol Details

3.1 Common Details

3.1.1 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 provided that their external behavior is consistent with what is described in this document.

The MSMQ: Directory Service Change Notification Protocol is used to notify to queue managers of changes in their queue and machine objects within a directory service. A directory service maintains a collection of objects that are specified in section 3.1.1.1. Queue and machine objects are subsets of the objects contained in a directory service. Queue managers maintain only queue and machine objects. Section 3.1.1.2 specifies the abstract data model that MAY be needed to maintain these objects.

3.1.1.1 Directory Service Abstract Data Model

Directory service objects, directory service object properties, and directory service object relationships are as specified in the [MS-MQDS] Abstract Data Model.

3.1.1.2 Queue Manager Abstract Data Model

Queue managers MUST maintain the elements discussed in the following sections.

3.1.1.2.1 Notification Queue

A local private notification queue. The name for this queue MUST be in the format specified by the following ABNF rule.

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

Queue managers MAY maintain the elements described in the Queue Objects table (section 3.1.1.2.2) and the Machine Objects table (section 3.1.1.2.3).

3.1.1.2.2 Queue Objects Table

A table of queue objects deployed at the queue manager, indexed by Instance Identifier. Each entry MAY contain:

- **Instance Identifier**, as specified in PROPID_Q_INSTANCE.
- **Type**, as specified in PROPID_Q_TYPE.
- **Base Priority**, as specified in PROPID_Q_BASEPRIORITY.
- **Journal**, as specified in PROPID_Q_JOURNAL.

- **Quota**, as specified in PROPID_Q_QUOTA.
- **Journal Quota**, as specified in PROPID_Q_JOURNAL_QUOTA.
- **Create Time**, as specified in PROPID_Q_CREATE_TIME.
- **Modify Time**, as specified in PROPID_Q_MODIFY_TIME.
- **Security**, as specified in PROPID_Q_SECURITY.
- **Path Name**, as specified in PROPID_Q_PATHNAME.
- **Label**, as specified in PROPID_Q_LABEL.
- **Authenticate**, as specified in PROPID_Q_AUTHENTICATE.
- **Privacy Level**, as specified in PROPID_Q_PRIV_LEVEL.
- **Transaction**, as specified in PROPID_Q_TRANSACTION.
- **Timestamp**, as specified in PROPID_Q_TIMESTAMP.
- **System Queue**, as specified in PROPID_Q_SYSTEMQUEUE.
- **Multicast Address**, as specified in PROPID_Q_MULTICAST_ADDRESS.
- **Active Directory Service** path, as specified in PROPID_Q_ADS_PATH.

Each entry MAY contain additional queue properties that are specified in [MS-MQMQ] section 2.3.1.

3.1.1.2.3 Machine Objects Table

A table of machine objects deployed at the queue manager, indexed by name. Each entry MAY contain:

- **Instance Identifier**, as defined in PROPID_QM_MACHINE_ID.
- **Foreign**, as defined in PROPID_QM_FOREIGN.
- **Quota**, as defined in PROPID_QM_QUOTA.
- **Journal Quota**, as defined in PROPID_QM_JOURNAL_QUOTA.
- **Security**, as defined in PROPID_QM_SECURITY.

Each entry MAY contain additional machine properties that are specified in [MS-MQMQ] section 2.3.2.

3.1.2 Timers

The MSMQ: Directory Service Change Notification Protocol does not require protocol timers beyond those used internally by either the message queuing infrastructure or the RPC protocol. For more details, see [MS-MQMA] and [MS-RPCE].

3.1.3 Initialization

Each queue manager service SHOULD create a local private notification queue.<11> The name for this queue MUST be in the format specified by the following ABNF rule.

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

3.1.4 Higher-Layer Triggered Events

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

- The queue manager service is started.
- The queue manager gets a request to create, update, or delete a queue object, or to update a machine object.<12><13>
- The directory service gets a request to create, update, or delete a queue object, or to update a machine object.<14><15>
- The queue manager receives a change notification message.

3.1.4.1 Queue Manager Service Is Started

On startup, the queue manager service SHOULD open the local private notification queue to receive change notification messages.<16> The **QUEUE_FORMAT** used to open this queue MUST have a **QUEUE_FORMAT_TYPE** of either **QUEUE_FORMAT_TYPE_PRIVATE** for **dependent clients** or **QUEUE_FORMAT_TYPE_DIRECT** for **independent clients**. Information on dependent clients and independent clients is as specified in [MS-MQMA] and [MS-MQMQ].

3.1.4.2 A Queue Manager Is Alerted About the Creation, Update, or Deletion of a Queue Object; or About an Update of a Machine Object in the Directory Service

After an application creates, updates, or deletes a queue object, or updates a machine object in the directory service, <17> the local queue manager MAY<18> be alerted about these changes. If the local queue manager is the owner of that object, then it MUST update its internal state; otherwise, it MUST use the MSMQ: Directory Service Change Notification Protocol to send a change notification message to the owner of the object. On reception, the owner MUST update its internal state.<19>

The MSMQ: Directory Service Change Notification Protocol starts after the local queue manager is alerted of the change on the directory service. The mechanism by which the local queue manager is alerted about changes in the directory service by the application is protocol-independent and implementation-specific. The application MAY use a local RPC call to alert the local queue manager after it changes a queue or a machine object on the directory service. An alternative MAY be to add functionality to the implementation of the LDAP [RFC2251] [MS-ADTS] pass-through client to alert the local queue manager when calls to change these objects are made. The implementer SHOULD decide what mechanism better meets the requirements.

The mechanism as to how a queue manager updates its internal state does not influence the behavior or specification of the MSMQ: Directory Service Change Notification Protocol. In addition, this protocol is independent of the state that is maintained in a queue manager; however, this protocol specifies the abstract model of what a queue manager MAY<20> maintain in section 3.1.1.

The mechanism as to how a queue manager sends a change notification message to the owner of an object is specified in section 3.1.5.1.1.

3.1.4.3 The Directory Service Gets a Request to Create, Update, or Delete a Queue Object, or to Update a Machine Object

After an application creates, updates, or deletes a queue object, or updates a machine object in the directory service, <21> the directory service MAY<22> use the MSMQ: Directory Service Change Notification Protocol to send a change notification message to the owner of the object. On reception, the owner of the object MUST update its internal state.<23>

The mechanism as to how a directory service sends a change notification message to the owner of an object is specified in section 3.1.5.1.2.

3.1.4.4 The Queue Manager Receives a Change Notification Message

The queue manager MUST be listening for incoming change notification messages on its local private notification queue.<24> On reception, the queue manager MAY decide to update its internal state immediately or defer the update for a later time.<25><26> It MAY be a best practice to update change notification messages sent from directory services immediately and to schedule change notification messages sent from queue managers in batches to be executed at designated time intervals. The protocol treats directory services as trusted and queue managers as untrusted. Because queue managers may not be trusted, batching the notification processing MAY<27> prevent denial of service attacks. For security, see section 5.

The mechanism as to how a queue manager receives a change notification messages is specified in section 3.1.5.2.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Sending a Change Notification Message

In the MSMQ: Directory Service Change Notification Protocol, a change notification message can be sent either by a directory service or by a queue manager.<28> This section specifies sending a change notification message from a queue manager (see section 3.1.5.1.1) and sending a change notification message from a directory aervice (see section 3.1.5.1.2). Finally, section 3.1.5.1.3 specifies sending a notification message within an MSMQ message. Sections 3.1.4.2 and 3.1.4.3 specify the details that trigger a message to be sent.

3.1.5.1.1 Sending a Change Notification Message from a Queue Manager

The queue manager MUST create a Notification Body and a Notification Header. The Notification Body MUST be appended to the Notification Header, and both MUST be attached to a message. Finally, the message MUST be sent through MSMQ to the destination queue manager.

The Notification Body, which is specified in section 2.2.9, MUST be created as follows. It MUST be set to a Unicode string in the format specified by the ABNF rule in section 2.2.3. *Event* MUST contain **0** if there is no change notification event;<29> or **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. *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 Notification Header, which is specified in section 2.2.8, MUST be created as follows. Field *Version* MUST be set to 0x02 to indicate that a change notification message is originated by a queue manager. 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 Notification Header 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 3.1.5.1.3.

3.1.5.1.2 Sending a Change Notification Message from a Directory Service

The directory service MUST create Notification Update instances and a Notification Header. The Notification Update instances MUST be appended to the Notification Header, and the result MUST be attached to a message. Finally, the message MUST be sent through MSMQ to the destination queue manager.

The Notification Update instance, which is specified in section 2.2.10, 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 **PropertyId[]** MUST be filled as follows:

    PropertyId[0] = PROPID_Q_TYPE

    PropertyId[2] = PROPID_Q_INSTANCE

    PropertyId[3] = PROPID_Q_BASEPRIORITY

    PropertyId[4] = PROPID_Q_JOURNAL

    PropertyId[5] = PROPID_Q_QUOTA

    PropertyId[6] = PROPID_Q_JOURNAL_QUOTA

    PropertyId[7] = PROPID_Q_CREATE_TIME

    PropertyId[8] = PROPID_Q_MODIFY_TIME

PropertyId[9] = PROPID_Q_SECURITY

PropertyId[10] = PROPID_Q_PATHNAME

PropertyId[11] = PROPID_Q_LABEL

PropertyId[12] = PROPID_Q_AUTHENTICATE

PropertyId[13] = PROPID_Q_PRIV_LEVEL

PropertyId[14] = PROPID_Q_TRANSACTION

These properties MUST be included also.

PROPID_Q_TIMESTAMP

PROPID_Q_SYSTEMQUEUE

PROPID_Q_MULTICAST_ADDRESS

These properties MUST NOT be included. The receiving queue manager MUST ignore these properties if received.

PROPID_Q_ADS_PATH

- If **Command** is set to 0x01 (updated), and the modified object is a queue object, the **PropertyId[]** MUST be filled as follows:

  PropertyId[0] = 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 **PropertyId[]** MUST be filled as follows:

  PropertyId[0] = PROPID_QM_MACHINE_ID

  PropertyId[1] = 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 are properties that MAY be modified.

PROPID_QM_QUOTA

PROPID_QM_JOURNAL_QUOTA

PROPID_QM_SECURITY

- 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 Notification Header, which is described in section 2.2.8, MUST be created as follows. Field **Version** MUST be set to 0x01 to indicate that a change notification message is originated by a directory service. Field **NumberOfUpdateNotifications** MUST be set to the number of Notification Update instances to be included in field **Data**. A directory service MAY 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 Notification Header 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 3.1.5.1.3.

3.1.5.1.3 Sending a Change Notification Message Within a Message Queuing Message

The Notification Header MUST be sent within a message instance through MSMQ to the destination queue manager. Details on sending messages through MSMQ are contained in the Message Queuing (MSMQ): Message Queuing Binary Protocol, as specified in [MS-MQQB]. This section specifies the values that MUST be set that are specific to notification messages. For fields not specified here, values MUST be used as specified in [MS-MQQB]. The structures and fields referenced in the following bullets are specified in [MS-MQQB]; see BaseHeader, MessagePropertiesHeader, SecurityHeader, UserHeader, and UserMessage Packet.

- Priority MUST be set to normal. The value for **BaseHeader.Flags.PR** MUST be set to 0x3.
- Type of message MUST be set to 0x1, which indicates a UserMessage Packet. The value for **BaseHeader.Flags.IN** MUST be set to 0x1.

- Time to reach queue MUST be set to five minutes. The value for **BaseHeader.TimeToReachQueue** MUST be set to 0x0000012C.
- Delivery mode of the packet MUST be set to express messaging. The value for **UserHeader.Flags.DM** MUST be set to 0x0.
- Type of destination queue MUST be set to private queue. The value for **UserHeader.Flags.DQ** MUST be set to 0x6.
- Destination queue MUST be set with the ID of the notification queue. The field **UserHeader.DestinationQueue** MUST be set to 0x00000003.
- Message class MUST be set to normal. The field **MessagePropertiesHeader.MessageClass** MUST be set to MQMSG_CLASS_NORMAL.
- The sender MUST specify a hash algorithm. The field **MessagePropertiesHeader.HashAlgorithm** MUST be set.<30>

If the message is being sent from a directory service, the MSMQ message MUST be signed according to the following rules.

- The security ID MUST contain the **GUID** of the queue manager that is local to the directory service. The field **SecurityHeader.Flags.ST** MUST be set to 0x2.
- The message MUST be authenticated. The field **SecurityHeader.Flags.AU** MUST be set to the binary 1.

  The **SecurityData.Signature** field MUST be set by the sender, and the receiver MUST authenticate the packet by verifying the **SecurityData.Signature** field.

  Sender: The sender MUST compute a hash of the fields specified by the **Flags.AS** field using the hash algorithm specified by the **MessagePropertiesHeader.HashAlgorithm** field.

  Receiver: The receiver MUST authenticate the signature by re-computing the hash and comparing it to the decrypted signature value. If the values are not identical, the packet MUST be ignored.
- The default cryptographic provider MUST be used. The field **SecurityHeader.Flags.DE** MUST be set.
- The security header MUST contain a security data field. The field **SecurityHeader.Flags.AI** MUST be set.
- The sender MUST indicate the authentication signature type. The field **SecurityHeader.Flags.AS** MUST be set to 0x1.
- The sender MUST specify the size of the sender ID. The file **SecurityHeader.SenderIdSize** MUST be set to the size in bytes of the security identifier in the **SecurityData.SecurityID** field, which is the size of a **GUID** type instance.
- The sender MUST specify the size of the signature. The field **SecurityHeader.SignatureSize** MUST be set to the size in byes of the sender certificate in the **SecurityData.Signature** field, which is 0x0080 (128).
- The following additional security information MUST be provided:
  - The field **SecurityHeader.SecurityData.SecurityID** MUST be set with the **GUID** of the queue manager residing in the same machine as the directory service.

- The field **SecurityHeader.SecurityData.Signature** MUST be set according to the values specified in the **SecurityHeader.Flags.AS** field, and the hash algorithm is specified by the **MessagePropertiesHeader.HashAlgorithm** field.

3.1.5.2 Receiving a Change Notification Message

In the MSMQ: Directory Service Change Notification Protocol, a change notification message MUST be received by the queue manager that is the owner of the object being changed. The change notification message SHOULD<31> be sent either from a directory service or a queue manager.<32> Section 3.1.4.4 specifies details that cause a message to be received.

On reception of a change notification message, the queue manager MUST verify the value of field **Version** in the Notification Header. If the value of field **Version** is 0x00, field **Data** MUST be interpreted as an array of instances of Notification Update. The number of instances is determined by the value of field **NumberOfUpdateNotifications**. In this case, the message sender is a directory service. For receiving a change notification message from a directory service, see section 3.1.5.2.1.

Otherwise, if the value of field **Version** is 0x01, the field **Data** MUST be interpreted as an instance of Notification Body. In this case, the message sender is a queue manager. For receiving a change notification message from a queue manager, see section 3.1.5.1.2.

3.1.5.2.1 Receiving a Change Notification Message from a Directory Service

The data structure for a message sent from a directory service is a Notification Update. In this case, the queue manager SHOULD update the state of the queue or machine objects referenced by **PathName** or **GuidIdentifier** within each instance of Notification Update. The queue manager MAY use the property identifiers and values for the updating process. As an alternative, the queue manager MAY request information from the directory service specified in **GuidMasterId**. If the queue manager uses the property identifiers and values sent within each Notification Update instance, it MUST verify the signature of the notification message.

The object type (either queue object or machine object) in a Notification Update MUST be determined according to the following rules:

- If **Command** is set to 0x00 (created) or 0x01 (updated), the object MUST be one of the following:
  - If the value of **PropertyId[0]** is greater than 1,000, the object type is (**PropertyId[0]** - 1,000) / 100.
  - Otherwise, the object type is **PropertyId[0]** / 100.

  The result MUST always be either 1 or 2 where 1 represents a queue object, and 2 represents a machine object.
- If **Command** is set to 0x02 (deleted), the object type is the value of **PropertyValue[1]**, which MUST be MQDS_QUEUE.

Section 3.1.1 specified the abstract model that MAY be used as a reference to maintain data specific to queue and machine objects in the queue manager.

3.1.5.2.2 Receiving a Change Notification Message from a Queue Manager

The data structure for a message sent from a queue manager is a Notification Body. In this case, the queue manager SHOULD 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 SHOULD request information from the directory service specified within **DomainController** for the updating process.

Section 3.1.1 describes the abstract model that MAY be used as a reference to maintain data specific to queue and machine objects in the queue manager.

3.1.6 Timer Events

There are no timer events.

3.1.7 Other Local Events

There are no other local events.

4 Protocol Examples

The following sections describe operations as used in common scenarios to illustrate the function of the MSMQ: Directory Service Change Notification Protocol.

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 independent clients. Both independent clients 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.

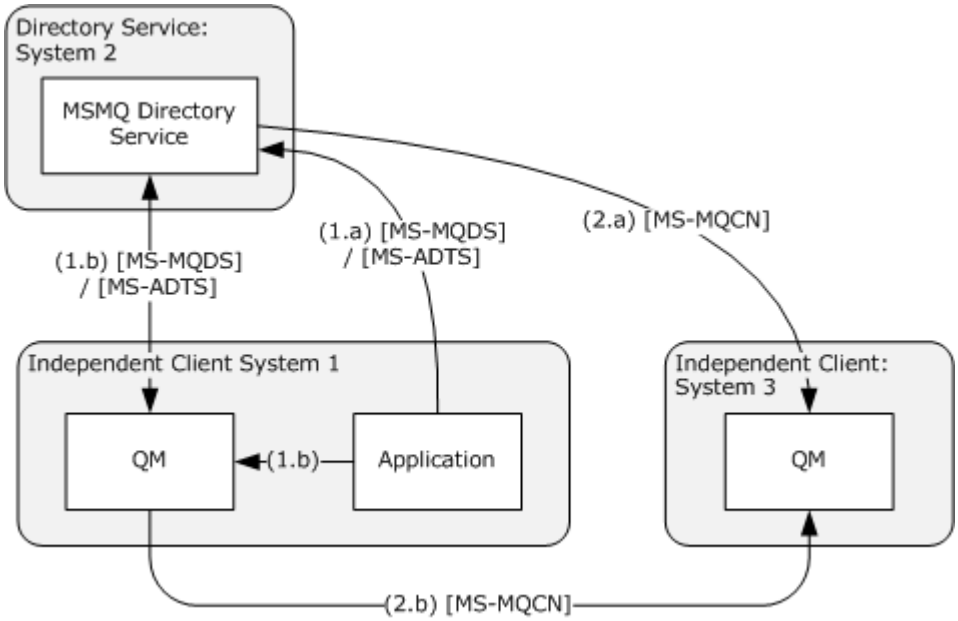


Figure 1: Creation of a public queue on a remote queue manager

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.

   - a. 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.

   - b. 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.

- a. 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.
- b. 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

The following sections specify security considerations for implementers of the MSMQ: Directory Service Change Notification Protocol.

5.1 Security Considerations for Implementers

This protocol MAY allow denial of service 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.<33>

5.2 Index of Security Parameters

There are no security parameters for this protocol.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista
- Windows Vista SP1
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription, and the term MAY implies that Windows 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<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, and Windows Vista, queue managers process notification messages as soon as they arrive for messages from directory service, and then schedule notification messages to be processed at specific time intervals for messages from queue managers. Notification messages from queue managers are processed at a maximum rate of once per 15 minutes to prevent denial of service attacks.

<4> Section 1.7: On Windows NT and Windows 2000, queue managers process messages only from directory service. Other messages are ignored. On Windows XP, Windows Server 2003, and Windows Vista, queue managers process messages either from directory service or from queue managers. Other messages are ignored.

<5> Section 2.2.8: 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<6> Section 2.2.8: On all Windows versions, setting *Version* to an invalid value MAY generate a trace entry in the recipient.

<7> Section 2.2.9: On all Windows versions, setting *Event* to an invalid value MAY generate a trace entry in the recipient.

<8> Section 2.2.10: *PreviousSequenceNumber*, *SequenceNumber*, and *PurgeSequenceNumber* are not used in any Windows implementation of MSMQ. These are ignored when receiving change notification messages.

<9> Section 2.2.10: *PreviousSequenceNumber*, *SequenceNumber*, and *PurgeSequenceNumber* are not used in any Windows implementation of MSMQ. These are ignored when receiving change notification messages.

<10> Section 2.2.10: *PreviousSequenceNumber*, *SequenceNumber*, and *PurgeSequenceNumber* are not used in any Windows implementation of MSMQ. These are ignored when receiving change notification messages.

<11> Section 3.1.3: On Windows Server 2008 and Windows Vista SP1, when a queue manager is running in **Workgroup mode**, the notification queue is not created.

<12> Section 3.1.4: 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<13> Section 3.1.4: On Windows NT and Windows 2000, queue managers process messages only from directory services . Other messages are ignored. On Windows XP, Windows Server 2003, and Windows Vista, queue managers process messages either from directory services or from queue managers. Other messages are ignored.

<14> Section 3.1.4: 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<15> Section 3.1.4: On Windows NT and Windows 2000, queue managers process messages only from directory services. Other messages are ignored. On Windows XP, Windows Server 2003, and Windows Vista, queue managers process messages either from directory services or from queue managers. Other messages are ignored.

<16> Section 3.1.4.1: On Windows Server 2008 and Windows Vista SP1, when a queue manager is running in Workgroup mode, the notification queue is not opened.

<17> Section 3.1.4.2: On Windows NT, MSMQ uses an implementation of the Message Queuing (MSMQ): Directory Service Protocol, as specified in [MS-MQDS]. On Windows 2000, MSMQ uses the Lightweight Directory Access Protocol (LDAP), as specified in [RFC2251] and in [MS-ADTS] if available; otherwise, it uses the Message Queuing (MSMQ): Directory Service Protocol, as specified in [MS-MQDS]. On Windows XP, Windows Server 2003, and Windows Vista, MSMQ always uses LDAP. On Windows Vista, the client side of the Message Queuing (MSMQ): Directory Service Protocol is removed from MSMQ, but it still serves MSMQ: Directory Service Protocol clients from MSMQ Version 2 and MSMQ Version 3; see [MS-MQMA].

<18> Section 3.1.4.2: 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol .

<19> Section 3.1.4.2: On Windows NT and Windows 2000, queue managers process notification messages as soon as they arrive. In contrast, on Windows XP, Windows Server 2003, and Windows Vista, queue managers process notification messages as soon as they arrive for messages from directory services and schedule notification messages to be processed at specific time intervals for messages from queue managers. Notification messages from queue managers are processed at a maximum rate of once per 15 minutes to prevent denial of service attacks.

<20> Section 3.1.4.2: On all Windows versions, queue managers maintain a list of queue objects and their properties. In addition, queue managers maintain properties of the related machine objects.

<21> Section 3.1.4.3: On Windows NT, MSMQ uses an implementation of the Message Queuing (MSMQ): Directory Service Protocol, as specified in [MS-MQDS]. On Windows 2000, MSMQ uses the Lightweight Directory Access Protocol (LDAP) [RFC2251] [MS-ADTS] if available; otherwise, it uses the Message Queuing (MSMQ): Directory Service Protocol, as specified in [MS-MQDS]. On Windows XP, Windows Server 2003, and Windows Vista, MSMQ always uses LDAP. On Windows Vista, the client side of the MSMQ: Directory Service Protocol is removed from MSMQ, but it still serves MSMQ: Directory Service Protocol clients from MSMQ Version 2 and MSMQ Version 3; see [MS-MQMA].

<22> Section 3.1.4.3: 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<23> Section 3.1.4.3: On Windows NT and Windows 2000, queue managers process notification messages as soon as they arrive. In contrast, on Windows XP, Windows Server 2003, and Windows Vista, queue managers process notification messages as soon as they arrive for messages from directory services and schedule notification messages to be processed at specific time intervals for messages from queue managers. Notification messages from queue managers are processed at a maximum rate of once per 15 minutes to prevent denial of service attacks.

<24> Section 3.1.4.4: On Windows Server 2008 and Windows Vista SP1, when a queue manager is running in Workgroup mode, the notification queue is not opened.

<25> Section 3.1.4.4: On Windows NT and Windows 2000, queue managers process notification messages as soon as they arrive. In contrast, on Windows XP, Windows Server 2003, and Windows Vista, queue managers process notification messages as soon as they arrive for messages from directory services and schedule notification messages to be processed at specific time intervals for messages from queue managers. Notification messages from queue managers are processed at a maximum rate of once per 15 minutes to prevent denial of service attacks.

<26> Section 3.1.4.4: On Windows Server 2003, Windows Vista, Windows Server 2008 and Windows Vista SP1, when a queue manager is running in Workgroup mode, change notification messages are ignored.

<27> Section 3.1.4.4: On Windows NT and Windows 2000, queue managers process notification messages as soon as they arrive. In contrast, on Windows XP, Windows Server 2003, and Windows Vista, queue managers process notification messages as soon as they arrive for messages from directory services and schedule notification messages to be processed at specific time intervals for messages from queue managers. Notification messages from queue managers are processed at a maximum rate of once per 15 minutes to prevent denial of service attacks.

<28> Section 3.1.5.1: 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<29> Section 3.1.5.1.1: On all Windows versions, if a message is received and the value of *Event* is 0, then the message is ignored.

<30> Section 3.1.5.1.3: MSMQ versions 1 and 2 use MD5, as specified in [RFC1321]; versions 2 and 4 use SHA1 as specified in [RFC3110].

<31> Section 3.1.5.2: On Windows Server 2003, Windows Vista, Windows Server 2008 and Windows Vista SP1, when a queue manager is running in Workgroup mode, change notification messages are ignored.

<32> Section 3.1.5.2: 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, and Windows Vista, the local queue manager notifies the destination queue manager via the MSMQ: Directory Service Change Notification Protocol.

<33> 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, and Windows Vista, queue managers process notification messages as soon as they arrive for messages from Directory Service and schedule notification messages to be processed at specific time intervals for messages from queue managers. Notification messages from queue managers are processed at a maximum rate of once per 15 minutes to prevent denial of service attacks.

7 Index

T

V

W