[MS-MQMQ-Diff]:

Message Queuing (MSMQ): Data Structures

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
5/11/2007	0.1	New	Version 0.1 release
8/10/2007	1.0	Major	Updated and revised the technical content.
9/28/2007	1.0.1	Editorial	Changed language and formatting in the technical content.
10/23/2007	1.0.2	Editorial	Changed language and formatting in the technical content.
11/30/2007	1.0.3	Editorial	Changed language and formatting in the technical content.
1/25/2008	1.0.4	Editorial	Changed language and formatting in the technical content.
3/14/2008	2.0	Major	Updated and revised the technical content.
5/16/2008	2.0.1	Editorial	Changed language and formatting in the technical content.
6/20/2008	3.0	Major	Updated and revised the technical content.
7/25/2008	3.0.1	Editorial	Changed language and formatting in the technical content.
8/29/2008	4.0	Major	Updated and revised the technical content.
10/24/2008	5.0	Major	Updated and revised the technical content.
12/5/2008	6.0	Major	Updated and revised the technical content.
1/16/2009	6.1	Minor	Clarified the meaning of the technical content.
2/27/2009	7.0	Major	Updated and revised the technical content.
4/10/2009	7.1	Minor	Clarified the meaning of the technical content.
5/22/2009	7.2	Minor	Clarified the meaning of the technical content.
7/2/2009	7.3	Minor	Clarified the meaning of the technical content.
8/14/2009	7.4	Minor	Clarified the meaning of the technical content.
9/25/2009	8.0	Major	Updated and revised the technical content.
11/6/2009	9.0	Major	Updated and revised the technical content.
12/18/2009	10.0	Major	Updated and revised the technical content.
1/29/2010	11.0	Major	Updated and revised the technical content.
3/12/2010	11.1	Minor	Clarified the meaning of the technical content.
4/23/2010	11.1.1	Editorial	Changed language and formatting in the technical content.
6/4/2010	11.2	Minor	Clarified the meaning of the technical content.
7/16/2010	12.0	Major	Updated and revised the technical content.
8/27/2010	13.0	Major	Updated and revised the technical content.
10/8/2010	13.1	Minor	Clarified the meaning of the technical content.
11/19/2010	14.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
1/7/2011	15.0	Major	Updated and revised the technical content.
2/11/2011	16.0	Major	Updated and revised the technical content.
3/25/2011	17.0	Major	Updated and revised the technical content.
5/6/2011	17.1	Minor	Clarified the meaning of the technical content.
6/17/2011	17.2	Minor	Clarified the meaning of the technical content.
9/23/2011	18.0	Major	Updated and revised the technical content.
12/16/2011	19.0	Major	Updated and revised the technical content.
3/30/2012	19.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	19.1	Minor	Clarified the meaning of the technical content.
10/25/2012	19.2	Minor	Clarified the meaning of the technical content.
1/31/2013	19.2	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	20.0	Major	Updated and revised the technical content.
11/14/2013	20.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	20.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	20.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	21.0	Major	Significantly changed the technical content.
10/16/2015	21.0	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	21.0	None	No changes to the meaning, language, or formatting of the technical content.
6/1/2017	21.0	None	No changes to the meaning, language, or formatting of the technical content.
9/15/2017	22.0	Major	Significantly changed the technical content.
9/12/2018	23.0	Major	Significantly changed the technical content.
4/7/2021	24.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

Message Queuing (MSMQ): Data Structures contains common definitions and data structures that are used in various protocols in the set of Microsoft Message Queuing protocols. The documentation for individual protocols contains references to this document, as needed.

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

Active Directory: The Windows implementation of a general-purpose directory service, which uses LDAP as its primary access protocol. Active Directory stores information about a variety of objects in the network such as user accounts, computer accounts, groups, and all related credential information used by Kerberos [MS-KILE]. Active Directory is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS), which are both described in [MS-ADOD]: Active Directory Protocols Overview.

active queue: A queue that contains messages or is currently opened by an application. Active queues can be public queues, private queues, or outgoing queues.

administration queue: A messaging queue that receives Message Queuing (MSMQ) system-generated acknowledgment messages. An administration queue is available to MSMQ applications for checking message status.

backup site controller (BSC): An MSMQ Directory Service role played by an MSMQ queue manager. A BSC contains a read-only copy of the directory for a site. A BSC can satisfy directory lookup requests but cannot satisfy directory change requests. There can be zero or more BSCs in a site.

connected network: A network of computers in which any two computers can communicate directly through a common transport protocol (for example, TCP/IP or SPX/IPX). A computer can belong to multiple connected networks.

connector queue: A queue used by a connector server. Messages sent to foreign queues are temporarily stored in a connector queue before they are retrieved by the connector application.

connector server: A Message Queuing routing server that is configured to send messages between a Message Queuing site and one or more foreign sites. A connector server has a connector application running on it and two connector queues for each foreign site: one used for transactional messages and one used for nontransactional messages.

cursor: A data structure providing sequential access over a message queue. A cursor has a current pointer that lies between the head and tail pointer of the queue. The pointer can be moved forward or backward through an operation on the cursor (Next). A message at the current pointer can be accessed through a nondestructive read (Peek) operation or a destructive read (Receive) operation.

dead-letter queue: A queue that contains messages that were sent from a host with a request for negative source journaling and that could not be delivered. Message Queuing provides a transactional dead-letter queue and a non-transactional dead-letter queue.

direct format name: A name that is used to reference a public queue or a private queue without accessing the MSMQ Directory Service. Message Queuing can use the physical, explicit location information provided by direct format names to send messages directly to their destinations. For more information, see [MS-MQMQ] section 2.1.

directory service (DS): 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].

distinguished name (DN): A name that uniquely identifies an object by using the relative distinguished name (RDN) for the object, and the names of container objects and domains that contain the object. The distinguished name (DN) identifies the object and its location in a tree.

distribution list: An Active Directory object that can contain explicit references only to destinations published in Active Directory; that is, to public queues, queue aliases, and other distribution lists, but not to private and URL-named queues.

Domain Name System (DNS): A hierarchical, distributed database that contains mappings of domain names to various types of data, such as IP addresses. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

enterprise: A unit of administration of a network of MSMQ queue managers. An enterprise consists of an MSMQ Directory Service, one or more connected networks, and one or more MSMQ sites.

foreign queue: A messaging queue that resides on a computer that does not run an MSMQ messaging application.

format name: A name that is used to reference a queue when making calls to API functions.

fully qualified domain name (FQDN): An unambiguous domain name that gives an absolute location in the Domain Name System's (DNS) hierarchy tree, as defined in [RFC1035] section 3.1 and [RFC2181] section 11.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the GUID. See also universally unique identifier (UUID).

Lightweight Directory Access Protocol (LDAP): The primary access protocol for Active Directory. Lightweight Directory Access Protocol (LDAP) is an industry-standard protocol, established by the Internet Engineering Task Force (IETF), which allows users to query and update information in a directory service (DS), as described in [MS-ADTS]. The Lightweight Directory Access Protocol can be either version 2 [RFC1777] or version 3 [RFC3377].

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

message: A data structure representing a unit of data transfer between distributed applications. A message has message properties, which may include message header properties, a message body property, and message trailer properties.

message body: A distinguished message property that represents the application payload.

message header: See message packet header.

message property: A data structure that contains a property identifier and a value, and that is associated with a message.

message queue: A data structure containing an ordered list of zero or more messages. A queue has a head and a tail and supports a first in, first out (FIFO) access pattern. Messages are

appended to the tail through a write operation (Send) that appends the message and increments the tail pointer. Messages are consumed from the head through a destructive read operation (Receive) that deletes the message and increments the head pointer. A message at the head can also be read through a nondestructive read operation (Peek).

Message Queuing Information Store (MQIS): The directory service store used by MSMQ Directory Service.

Message Transfer Protocol: A Message Transfer Protocol defines a mechanism for reliably transferring messages between two message queues located on two different hosts.

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

MSMQ 1.0 digital signature: A digital signature based on a hash of the MSMQ 1.0 Digital Signature Properties section in [MS-MQMQ]. This signature type is supported by all versions of Message Queuing.

MSMQ 2.0 digital signature: A digital signature that is more robust than the MSMQ 1.0 digital signature and is based on a hash of the MSMQ 2.0 Digital Signature Properties section in [MS-MQMQ]. This signature type is not supported by MSMQ version 1.

MSMQ 3.0 digital signature: A digital signature that is used only for messages sent to distribution lists or multiple-element format names and is based on a hash of the MSMQ 3.0 Digital Signature Properties section in [MS-MQMQ]. This signature type is not supported by MSMQ version 1 nor MSMQ version 2.

MSMQ Directory Service: A network directory service that provides directory information, including key distribution, to MSMQ. It initially shipped in the Windows NT 4.0 operating system Option Pack for Windows NT Server as part of MSMQ. This directory service predates and is superseded by Active Directory (AD).

MSMQ Directory Service server: An MSMQ queue manager that provides MSMQ Directory Service. The server can act in either of the MSMQ Directory Service roles: Primary Site Controller (PSC) or Backup Site Controller (BSC).

MSMQ mixed-mode: When upgrading from MSMQ 1.0 in Windows NT 4.0 operating system to MSMQ 2.0 in Windows 2000 operating system, a transitional mode known as mixed-mode environment is supported. Although not intended as a final deployment strategy, there is full support for this mixed-mode, which allows MSMQ 1.0 controller servers to coexist in the same enterprise with MSMQ 2.0 directory service servers, supporting both MSMQ 1.0 and MSMQ 2.0 directory service clients. In mixed-mode, the MSMQ replication service is used to synchronize MQIS with Active Directory (AD).

MSMQ queue manager: An MSMQ service hosted on a machine that provides queued messaging services. Queue managers manage queues deployed on the local computer and provide asynchronous transfer of messages to queues located on other computers. A queue manager is identified by a globally unique identifier (GUID).

MSMQ routing server: A role played by an MSMQ queue manager. An MSMQ routing server implements store and forward messaging. A routing server can provide connectivity between different connected networks within a site or can provide session concentration between sites.

MSMQ site: A network of computers, typically physically collocated, that have high connectivity as measured in terms of latency (low) and throughput (high). A site is represented by a site object in the directory service. An MSMQ site maps one-to-one with an Active Directory site when Active Directory provides directory services to MSMQ.

MSMQ supporting server: A role played by an MSMQ queue manager. An MSMQ supporting server supports applications to send and receive messages through the Message Queuing (MSMQ): Queue Manager Client Protocol [MS-MQMP].

nontransactional message: A message that is sent outside of a transaction.

NULL GUID: A GUID of all zeros.

order acknowledgment: A special acknowledgment message that is generated by a receiving queue manager to acknowledge receipt of a message in a transactional queue.

order queue: A messaging queue that is used to monitor the arrival order of messages that are sent as part of a transaction.

outgoing queue: A temporary internal queue that holds messages for a remote destination queue. The path name of an outgoing queue is identical to the path name of the corresponding destination queue. An outgoing queue is distinguished from its corresponding destination queue by the fact that the outgoing queue is located on the sending computer. The format name of an outgoing queue is identical to the format name used by the messages to reference the destination queue. Messages that reference the destination queue using a different format name are placed in a different outgoing queue.

path name: The name of the receiving computer where the messages for a particular queue are stored, and an optional PRIVATE$ key word indicating whether the queue is private, followed by the name of the queue. Path names can also refer to subqueues; for more information, see [MS-MQMQ] section 2.1.

Primary Enterprise Controller (PEC): An MSMQ Directory Service role played by an MSMQ queue manager. The PEC acts as the authority for the enterprise configuration information stored in the directory. There is only one PEC in an enterprise. The PEC also acts in the role of Primary Site Controller (PSC) for the site to which it belongs.

Primary Site Controller (PSC): An MSMQ Directory Service role played by an MSMQ queue manager. The PSC acts as the authority for the directory information for the site to which it belongs. The PSC can satisfy directory lookup requests and directory change requests. There is only one PSC per site.

private queue: An application-defined message queue that is not registered in the MSMQ Directory Service. A private queue is deployed on a particular queue manager.

property identifier: A DWORD value associated with an MSMQ object property that defines the property type and its semantic meaning.

public queue: An application-defined message queue that is registered in the MSMQ Directory Service. A public queue can be deployed at any queue manager.

queue: An object that holds messages passed between applications or messages passed between Message Queuing and applications. In general, applications can send messages to queues and read messages from queues.

queue journal: A queue that contains copies of the messages sent from a host when positive source journaling is requested.

queue manager (QM): A message queuing service that manages queues deployed on a computer. A queue manager can also provide asynchronous transfer of messages to queues deployed on other queue managers.

queue property: A data structure that contains a property identifier and a value, and is associated with a message queue.

quota: The physical disk quota for messages in the queue.

Remote Access Service (RAS) server: A type of network access server (NAS) that provides modem dial-up or virtual private network (VPN) access to a network.

remote procedure call (RPC): A communication protocol used primarily between client and server. The term has three definitions that are often used interchangeably: a runtime environment providing for communication facilities between computers (the RPC runtime); a set of request-and-response message exchanges between computers (the RPC exchange); and the single message from an RPC exchange (the RPC message). For more information, see [C706].

routing link: See MSMQ routing link.

security descriptor: A data structure containing the security information associated with a securable object. A security descriptor identifies an object's owner by its security identifier (SID). If access control is configured for the object, its security descriptor contains a discretionary access control list (DACL) with SIDs for the security principals who are allowed or denied access. Applications use this structure to set and query an object's security status. The security descriptor is used to guard access to an object as well as to control which type of auditing takes place when the object is accessed. The security descriptor format is specified in [MS-DTYP] section 2.4.6; a string representation of security descriptors, called SDDL, is specified in [MS-DTYP] section 2.5.1.

security identifier (SID): An identifier for security principals that is used to identify an account or a group. Conceptually, the SID is composed of an account authority portion (typically a domain) and a smaller integer representing an identity relative to the account authority, termed the relative identifier (RID). The SID format is specified in [MS-DTYP] section 2.4.2; a string representation of SIDs is specified in [MS-DTYP] section 2.4.2 and [MS-AZOD] section 1.1.1.2.

subqueue: A message queue that is logically associated, through a naming hierarchy, with a parent message queue. Subqueues can be used to partition messages within the queue. For example, a queue journal can be a subqueue that holds a copy of each message consumed from its parent queue.

system queue: An internal queue that is used by the queue manager for a purpose other than holding messages destined for a remote destination queue.

transactional message: A message sent as part of a transaction. Transaction messages must be sent to transactional queues.

transactional queue: A queue that contains only transactional messages.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The Unicode standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

unit of work: A set of individual operations that MSMQ must successfully complete before any of the individual MSMQ operations can be considered complete.

workgroup mode: A Message Queuing deployment mode in which the clients and servers operate without using a Directory Service. In this mode, features pertaining to message security, efficient routing, queue discovery, distribution lists, and aliases are not available. See also Directory-Integrated mode.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[FIPS180-2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-2, August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[FIPS197] FIPS PUBS, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[IANAIMA] IANA, "Internet Multicast Addresses", March 2007, http://www.iana.org/assignments/multicast-addresses

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

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

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

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

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

[MS-SAMR] Microsoft Corporation, "Security Account Manager (SAM) Remote Protocol (Client-to-Server)".

[PKCS5] RSA Laboratories, "PKCS #5: Password-Based Cryptography Standard", PKCS #5, Version 2.0, March 1999, http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-5-password-based-cryptography-standard.htm

[RFC1319] Kaliski, B., "The MD2 Message-Digest Algorithm", RFC 1319, April 1992, http://www.rfc-editor.org/rfc/rfc1319.txt

[RFC1320] Rivest, R., "The MD4 Message-Digest Algorithm", RFC 1320, April 1992, http://www.ietf.org/rfc/rfc1320.txt

[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

[RFC2268] Rivest, R., "A Description of the RC2(r) Encryption Algorithm", RFC 2268, March 1998, http://www.rfc-editor.org/rfc/rfc2268.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

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt

[RFC3447] Jonsson, J. and Kaliski, B., "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003, http://www.ietf.org/rfc/rfc3447.txt

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.rfc-editor.org/rfc/rfc3986.txt

[RFC4514] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006, http://www.ietf.org/rfc/rfc4514.txt

[RFC4516] Network Working Group, Smith, M., Ed., and Howes, T., "Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator", RFC 4516, June 2006, http://www.ietf.org/rfc/rfc4516.txt

[RFC4757] Jaganathan, K., Zhu, L., and Brezak, J., "The RC4-HMAC Kerberos Encryption Types Used by Microsoft Windows", RFC 4757, December 2006, http://www.ietf.org/rfc/rfc4757.txt

1.2.2 Informative References

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

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

[MSDN-ACP] Microsoft Corporation, "Microsoft AES Cryptographic Provider", http://msdn.microsoft.com/en-us/library/aa386979(VS.85).aspx

[MSDN-BCP] Microsoft Corporation, "Microsoft Base Cryptographic Provider", http://msdn.microsoft.com/en-us/library/aa386980(VS.85).aspx

[MSDN-CSP] Microsoft Corporation, "Cryptographic Provider Names", http://msdn.microsoft.com/en-us/library/aa380243.aspx

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

1.3 Structure Overview

The common definitions, naming formats, structures, data types, and error codes defined in this document are used by the member protocols of the Microsoft Message Queuing (MSMQ) protocol set.

1.4 Relationship to Protocols and Other Structures

The data types in this document are used by protocols in the set of Microsoft Message Queuing protocols.

1.5 Applicability Statement

None.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

Some structures in this document use **HRESULT** values as defined in [MS-ERREF] section 2.1. Vendors can define their own **HRESULT** values, provided that they set the C bit (0x20000000) for each vendor-defined value, indicating that the value is a customer code.

2 Definitions and Structures

This section discusses data structures that are used by various protocols in the set of Microsoft Message Queuing protocols.

2.1 MSMQ Queue Names

The following sections describe the various ways to designate Message Queuing queues.

2.1.1 Path Names

The path of a public queue consists of the name of the computer hosting the queue and the name of the queue separated by a backward slash in the form "ComputerName\QueueName". The names of private queues are prefixed by the string "private$" separated by a backward slash. Thus, the path of a private queues has the form "ComputerName\private$\QueueName". The names of system queues are prefixed by the string "system$" separated by a semicolon. Thus, the path of a system queue has the form "ComputerName\system$;QueueName".

A queue path name MUST conform to the following format in Augmented Backus-Naur Form (ABNF) notation.

```
QueuePathName = (Computer "\" QueueName /
                 Computer "\private$\" QueueName /
                 Computer "\system$;" QueueName)
Computer      = 1*256(VCHAR)VCHAR         = %x21-7E
```

For MSMQ 1.0, the format is as follows.

```
QueueName = 1*124(%x21-3A / %x3C-5B / %x5D-7f)
    ; Exclude backslash and semicolon
```

For MSMQ 2.0–MSMQ 3.0, the format is as follows.

```
QueueName = 1*124(%x21 / %x23-2A / %x2D-3A / %x3C-5B / %x5D-7f)
    ; Exclude backslash, semicolon, plus, comma, double quote
```

For MSMQ 4.0–MSMQ 6.0, the format is as follows.

```
QueueName = 1*124(%x21 / %x23-2A / %x2D-3A / %x3C-5B / %x5D-7f)
            [";" Subqueue]

Subqueue  = 1*32(%x21 / %x23-2A / %x2D-3A / %x3C-5B / %x5D-7f)
```

2.1.2 Direct Format Names

Direct format names are used to reference public or private queues without accessing the directory service. Message Queuing can use the information provided by direct format names to send messages directly to their destinations. Thus, direct format names can be used to send and receive messages in workgroup mode, send messages to computers on the Internet, and send messages directly to a computer.

A direct format name MUST conform to the following format in ABNF notation.

```
DirectName = PrivateQueuePath / PublicQueuePath / MachineQueuePath

PrivateQueuePath = "DIRECT=" Protocol ":"
                   ProtocolAddressSpecification "\PRIVATE$\"
                   QueueName [";JOURNAL"]

PublicQueuePath = "DIRECT=" Protocol ":"
                  ProtocolAddressSpecification "\" QueueName
                  [";JOURNAL"]

MachineQueuePath = "DIRECT=" Protocol ":"
                   ProtocolAddressSpecification "\SYSTEM$;"
                   ("JOURNAL" / "DEADLETTER" / "DEADXACT")

Subqueue=1*32(%x21 / %x23-2A / %x2D-3A / %x3C-5B / %x5D-7f)

Protocol = "TCP" / "OS" / "HTTP" / "HTTPS" / "IPX"
```

Where:

- <QueueName> is a queue path name from Path names (section 2.1.1).
- <ProtocolAddressSpecification> is the protocol-specific address format as defined in the following table.

Protocol	Description	Protocol address specification
TCP	Connection-oriented TCP over IP	Internet address notation (IP address)
IPX	Connection-oriented SPX over IPX	Network number and host number (separated by a colon ":" character)
OS	Connection using the native computer-naming convention	Any computer name supported by the underlying operating system
HTTP	Connection over HTTP	ProtocolAddressSpecification = "//" Host [":" Port] "/MSMQ" Port = *(%x30-39) Where: ▪ <Host> is either a computer name supported by the underlying operating system or an IP address. ▪ If <Port> is unspecified, a default of 80 is assumed.
HTTPS	Connection over HTTPS	ProtocolAddressSpecification = "//" Host [":" Port] "/MSMQ" Port = *(%x30-39) Where

Protocol	Description	Protocol address specification
		▪ <Host> is either a computer name supported by the underlying operating system or an IP address. ▪ If <Port> is unspecified, a default of 443 is assumed.

2.1.3 Public Format Names

Public format names and direct format names (section 2.1.2) are used to reference public queues. When a public format name is used, Message Queuing uses its internal routing algorithm to define the route to the destination queue.

Public format names contain the string "PUBLIC=" followed by the identifier assigned to the queue when it was created. This identifier is the GUID listed for the queue object in Active Directory.

A public format name MUST conform to the following format in ABNF notation.

```
PublicName = "PUBLIC=" QueueGuid

QueueGuid  = Guid

Guid       = 8HexDig %x2D 3(4HexDig %x2D) 12HexDig

HexDig     = Digit / "A" / "B" / "C" / "D"  / "E" / "F"

Digit      = %x30-39
```

2.1.4 Private Format Names

Private format names are used to reference private queues. When a private format name is used, Message Queuing uses its internal routing algorithm to define the route to the destination queue.

When Message Queuing detects a private format name, it does not refer to the directory service for information about the queue. However, it does use the directory service to look up information about the computer for routing purposes.

Private format names contain the string "PRIVATE=" followed by the identifier of the computer where the queue is registered and a hexadecimal number that identifies the queue.

A private format name MUST conform to the following format in ABNF notation.

```
PrivateName  = "PRIVATE=" ComputerGuid "\" 1*8HEXDIG [";JOURNAL"]

ComputerGuid = Guid
```

Where:

<Guid> is a **GUID**, as specified in section 2.1.3.

2.1.5 Distribution List Format Names

Distribution list format names are used to reference distribution lists (group objects) stored in Active Directory (as specified in [MS-ADTS]). Distribution list format names contain the string "DL=" followed by the distribution list identifier. This identifier is the GUID listed for the distribution list (group) object in Active Directory. The following is the general format used to reference a distribution list with optional inclusion of the Active Directory domain name.

The name MUST conform to the following format in ABNF notation.

```
DistributionListName = "DL=" DistributionListGuid ["@" DomainName]

DistributionListGuid = Guid

DomainName           = 1*255(VCHAR)

VCHAR                = %x21-7E
```

Where:

<Guid> is a **GUID**, as specified in section 2.1.3.

2.1.6 Machine, Connector, and Multicast Format Names

Machine format names are used to reference computer journals and dead-letter queues for a specific computer (for MSMQ 2.0 and MSMQ 3.0, Direct Format Names (section 2.1.2) also can be used for this purpose). Connector format names are used to reference the connector queues on a connector server. Multicast address format names (introduced in MSMQ 3.0) reference multiple destination queues that are addressed by an IPv4 Multicast address [IANAIMA].

These names MUST conform to the following format in ABNF notation.

```
Machine   = "MACHINE=" Guid

Connector = "CONNECTOR=" Guid

Multicast = "MULTICAST=" Address

Address   = dec-octet1 "." dec-octet "." dec-octet "." dec-octet ":" Port

dec-octet = Digit                ; 0-9
          / %x31-39 Digit        ; 10-99
          / "1" 2Digit           ; 100-199
          / "2" %x30-34 Digit    ; 200-249
          / "25" %x30-35         ; 250-255

dec-octet1 = "22" %x34-39        ; 224-229
           / "23" Digit          ; 230-239

Port      = Digit                ; 0-9
          / "1" Digit            ; 10-99
          / "1" 2Digit           ; 100-999
          / "1" 3Digit           ; 1000-9999
          / %x31-35 4Digit       ; 10000-59999
          / "6" %x30-34 3Digit   ; 60000-64999
          / "65" %x30-34 2Digit  ; 65000-65499
          / "655" %x30-32 Digit  ; 65500-65529
          / "6553" %x30-35       ; 65530-65535

Digit     = %x30-39
```

Where:

<Guid> is a GUID, as specified in section 2.1.3.

2.1.7 Multiple-Element Format Names

A multiple-element format name is formed as a concatenation of one or more public (section 2.1.3), private (section 2.1.4), direct (section 2.1.2), distribution list (section 2.1.5), connector (section 2.1.6), or multicast (section 2.1.6) format names, separated by commas. Thus, different kinds of format names used in Message Queuing can be used together as elements of a multiple-element format name.

The following example shows a multiple-element format name that contains a direct format name, a public format name, and a distribution list format name.

```
DIRECT=ComputerAddress\QueueName,PUBLIC=QueueGUID,DL=DL_GUID
```

Note A multiple-element format name containing an element that is a public, private, or distribution list format name cannot be used when there is no access to Active Directory.

Multiple-element format names cannot contain the format names of read-only queues, such as queue journals, computer journals, or dead-letter queues. An error is returned if the format name of a read-only queue is included in the multiple-element format name.

The name MUST conform to the following format in ABNF notation.

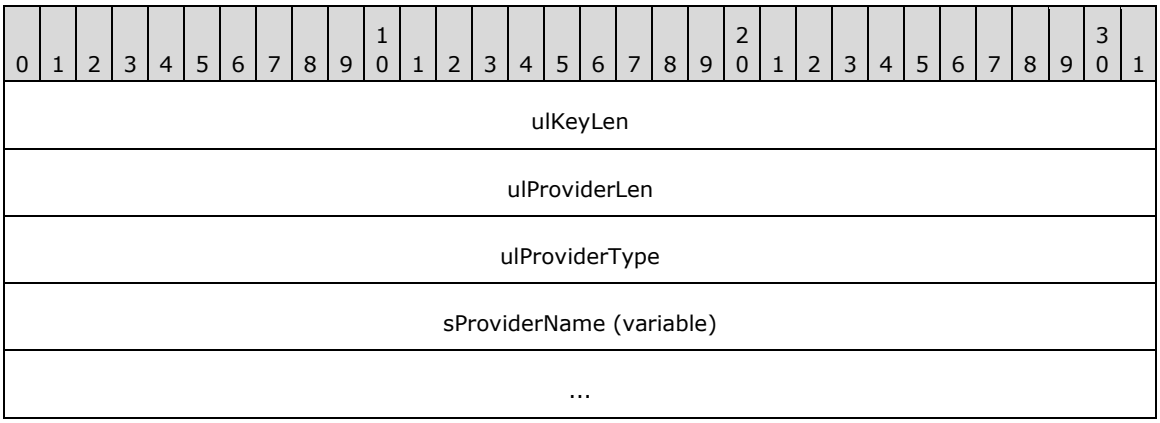
```
MultipleElementName = FormatName *("," FormatName)

FormatName = DirectName / PublicName / PrivateName /
             DistributionListName / Connector /
             Multicast
```

2.2 Structures

2.2.1 MQDSPUBLICKEY

The MQDSPUBLICKEY structure defines a public key certificate.



aBuf (variable)
...

ulKeyLen (4 bytes): An unsigned 32-bit integer that MUST contain the size, in bytes, of the data in the **aBuf** field.

ulProviderLen (4 bytes): An unsigned 32-bit integer that MUST contain the size, in bytes, of the provider name, including the terminating null character.

ulProviderType (4 bytes): An unsigned 32-bit integer that MUST contain an enumerated constant for the provider-type code. The value MUST be either PROV_RSA_FULL (0x00000001) or PROV_RSA_AES (0x00000018), indicating which provider was used to generate the public key certificate stored in the **aBuf** field.

sProviderName (variable): A null-terminated Unicode string that contains the provider name.

aBuf (variable): A buffer that MUST contain a **BLOBHEADER** ([MS-MQDS] section 2.2.19) structure, with the **aiKeyAlg** field set to CALG_RSA_KEYX (0x0000a400), followed by the public key certificate formatted as an **RSAPUBKEY** ([MS-MQDS] section 2.2.18) structure.

2.2.2 MQDSPUBLICKEYS

The MQDSPUBLICKEYS structure defines a set of MQDSPUBLICKEY (section 2.2.1) structures.

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
ulLen																															
cNumofKeys																															
aPublicKeys (variable)																															
...																															

ulLen (4 bytes): An unsigned 32-bit integer that MUST contain the size, in bytes, of the MQDSPUBLICKEYS structure.

cNumofKeys (4 bytes): An unsigned 32-bit integer that MUST contain the count of MQDSPUBLICKEY (section 2.2.1) structures in the array aPublicKeys.

aPublicKeys (variable): An array of MQDSPUBLICKEY (section 2.2.1) structures.

2.2.3 SECURITY_INFORMATION

A SECURITY_INFORMATION value applies to a SECURITY_DESCRIPTOR (as specified in [MS-DTYP] section 2.4.6). The value is constructed from zero or more bit flags from the following table.

Value	Meaning
OWNER_SECURITY_INFORMATION 0x00000001	Owner identifier of the object
GROUP_SECURITY_INFORMATION	Primary group identifier

Value	Meaning
0x00000002	
DACL_SECURITY_INFORMATION 0x00000004	Discretionary access control list (DACL) of the object
SACL_SECURITY_INFORMATION 0x00000008	System ACL of the object

This type is declared as follows:

```
typedef DWORD SECURITY_INFORMATION;
```

2.2.4 TA_ADDRESS

The TA_ADDRESS structure defines a single transport address of a specific type.

```
typedef struct _TA_ADDRESS {
  USHORT AddressLength;
  USHORT AddressType;
  UCHAR Address[1];
} TA_ADDRESS,
 *PTA_ADDRESS;
```

AddressLength: An unsigned 16-bit integer that MUST contain the size, in bytes, of the **Address** field. The value MUST be one of the following (by address type).

Address type prefix	Value
IP_	4
IPX_	10
FOREIGN_	16

AddressType: An unsigned 16-bit integer that MUST contain one of the values in the following table.

Value	Meaning
IP_ADDRESS_TYPE 0x0001	The **Address** field contains a 32-bit IP address.
IP_RAS_ADDRESS_TYPE 0x0002	The **Address** field contains a 32-bit IP address associated with a connection that is established through a Remote Access Service (RAS) server.
IPX_ADDRESS_TYPE 0x0003	The **Address** field contains a 4-byte netnum followed by a 6-byte nodenum. The netnum identifies the IPX network. The nodenum represents the IPX node address.
FOREIGN_ADDRESS_TYPE 0x0005	The **Address** field contains the GUID of a connected network object.

Address: The array of bytes that contains the address value.

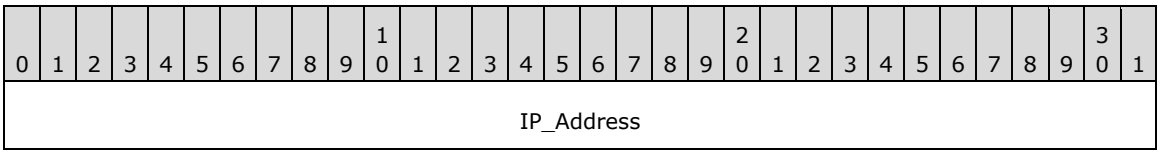
This MUST be one of the following:

- An IP address (as specified in section 2.2.4.1).

- An IPX address (as specified in section 2.2.4.2).
- A FOREIGN address is a **GUID** object (as specified in [MS-DTYP] section 2.3.4).

2.2.4.1 IP Address

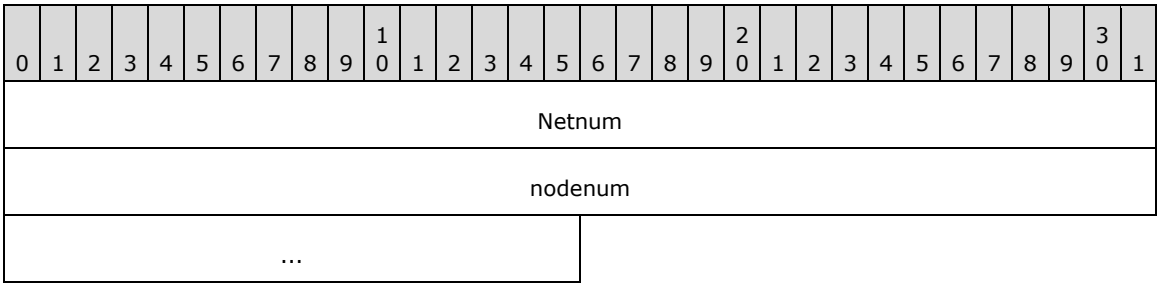
The IP Address packet is a numerical representation of an IPv4 address.



IP_Address (4 bytes): A 32-bit unsigned integer.

2.2.4.2 IPX Address

The IPX Address packet identifies a remote destination on a Novell Netware network.



2.2.5 SEQUENCE_INFO

The SEQUENCE_INFO structure stores the sequence information about the applicable message in a message stream sent from a given sending computer to a given destination queue.

```
typedef struct tagSEQUENCE_INFO {
  LONGLONG SeqID;
  ULONG SeqNo;
  ULONG PrevNo;
} SEQUENCE_INFO;
```

SeqID: Specifies a sequence identifier.

SeqNo: Specifies the sequence number of a message within the sequence identified by the **SeqID** member.

PrevNo: Specifies the sequence number of the message previous to the message indicated by the **SeqNo** member within the sequence identified by the **SeqID** member.

2.2.5.1 SEQUENCE_INFO (Packet)

The SEQUENCE_INFO (Packet) stores the sequence information about the applicable message in a message stream sent from a given sending computer to a given destination queue.

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
SeqID																															
...																															
SeqNo																															
PrevNo																															

SeqID (8 bytes): A 64-bit signed integer that specifies a sequence identifier.

SeqNo (4 bytes): A 32-bit unsigned integer that specifies the sequence number of a message within the sequence identified by the **SeqID** field.

PrevNo (4 bytes): A 32-bit unsigned integer that specifies the sequence number of the message previous to the message indicated by the **SeqNo** field within the sequence identified by the **SeqID** field.

2.2.6 QUEUE_FORMAT_TYPE

The QUEUE_FORMAT_TYPE enumeration identifies the type of name format being used.

```
typedef  enum __QUEUE_FORMAT_TYPE
{
  QUEUE_FORMAT_TYPE_UNKNOWN = 0,
  QUEUE_FORMAT_TYPE_PUBLIC = 1,
  QUEUE_FORMAT_TYPE_PRIVATE = 2,
  QUEUE_FORMAT_TYPE_DIRECT = 3,
  QUEUE_FORMAT_TYPE_MACHINE = 4,
  QUEUE_FORMAT_TYPE_CONNECTOR = 5,
  QUEUE_FORMAT_TYPE_DL = 6,
  QUEUE_FORMAT_TYPE_MULTICAST = 7,
  QUEUE_FORMAT_TYPE_SUBQUEUE = 8
} QUEUE_FORMAT_TYPE;
```

QUEUE_FORMAT_TYPE_UNKNOWN: The format type is unknown.

QUEUE_FORMAT_TYPE_PUBLIC: The QUEUE_FORMAT (section 2.2.7) structure contains a GUID (as specified in [MS-DTYP] section 2.3.4) that identifies a queue.

QUEUE_FORMAT_TYPE_PRIVATE: The QUEUE_FORMAT (section 2.2.7) structure contains an OBJECTID structure that identifies a queue.

QUEUE_FORMAT_TYPE_DIRECT: The QUEUE_FORMAT structure contains a direct format name string that identifies a queue.

QUEUE_FORMAT_TYPE_MACHINE: The QUEUE_FORMAT structure contains a GUID (as specified in [MS-DTYP] section 2.3.4) that identifies a queue.

QUEUE_FORMAT_TYPE_CONNECTOR: The QUEUE_FORMAT structure contains a GUID (as specified in [MS-DTYP] section 2.3.4) that identifies a connector queue. This is not supported by all protocols.

QUEUE_FORMAT_TYPE_DL: The QUEUE_FORMAT structure contains a GUID (as specified in [MS-DTYP] section 2.3.4) that identifies a distribution list (DL). This is not supported by all protocols.

QUEUE_FORMAT_TYPE_MULTICAST: The QUEUE_FORMAT structure contains a **MULTICAST_ID** (section 2.2.10) that identifies a multicast address. This is not supported by all protocols.

QUEUE_FORMAT_TYPE_SUBQUEUE: The QUEUE_FORMAT structure contains a direct name string that identifies a subqueue.

Note QUEUE_FORMAT_TYPE_SUBQUEUE was introduced in MSMQ version 4.

2.2.7 QUEUE_FORMAT

The QUEUE_FORMAT structure describes the type of queue being managed and an identifier for that queue.

```
typedef struct __QUEUE_FORMAT {
  unsigned char m_qft;
  unsigned char m_SuffixAndFlags;
  unsigned short m_reserved;
  [switch_is(m_qft)] union {
    [case(QUEUE_FORMAT_TYPE_UNKNOWN)]
            ;
    [case(QUEUE_FORMAT_TYPE_PUBLIC)]
      GUID m_gPublicID;
    [case(QUEUE_FORMAT_TYPE_PRIVATE)]
      OBJECTID m_oPrivateID;
    [case(QUEUE_FORMAT_TYPE_DIRECT)]
      [string] wchar_t* m_pDirectID;
    [case(QUEUE_FORMAT_TYPE_MACHINE)]
      GUID m_gMachineID;
    [case(QUEUE_FORMAT_TYPE_CONNECTOR)]
      GUID m_GConnectorID;
    [case(QUEUE_FORMAT_TYPE_DL)]
      DL_ID m_DlID;
    [case(QUEUE_FORMAT_TYPE_MULTICAST)]
      MULTICAST_ID m_MulticastID;
    [case(QUEUE_FORMAT_TYPE_SUBQUEUE)]
      [string] wchar_t* m_pDirectSubqueueID;
  };
} QUEUE_FORMAT;
```

m_qft: The type of queue format name. It MUST be set to one of the values of QUEUE_FORMAT_TYPE. It is used as a union discriminant in the QUEUE_FORMAT structure.

m_SuffixAndFlags: This member is broken into two subfields: **Suffix Type** is located in the 4 least-significant bits, and **Flags** is located in the 4 most-significant bits.

0	1	2	3	4	5	6	7
Flags				Suffix type			

Flags	Meaning
QUEUE_FORMAT_FLAG_NOT_SYSTEM 0x00	The specified queue is not a system queue.
QUEUE_FORMAT_FLAG_SYSTEM 0x80	The specified queue is a system queue.

Suffix type	Meaning
QUEUE_SUFFIX_TYPE_NONE 0x00	No suffix is specified. The **Flags** subfield MUST be set to 0x00. The **m_qft** member MUST NOT be set to 0x04.
QUEUE_SUFFIX_TYPE_JOURNAL 0x01	A journal suffix. The **Flags** subfield MUST be set to 0x80. The **m_qft** member MUST NOT be set to 0x05, 0x06, or 0x07.
QUEUE_SUFFIX_TYPE_DEADLETTER 0x02	A dead-letter suffix. The **Flags** subfield MUST be set to 0x80. The **m_qft** member MUST NOT be set to 0x01, 0x02, 0x05, 0x06, or 0x07.
QUEUE_SUFFIX_TYPE_DEADXACT 0x03	A transacted dead-letter suffix. The **Flags** subfield MUST be set to 0x80. The **m_qft** member MUST be set to 0x03 or 0x04.
QUEUE_SUFFIX_TYPE_XACTONLY 0x04	A transaction-only suffix. The **m_qft** member MUST be set to 0x05.
QUEUE_SUFFIX_TYPE_SUBQUEUE 0x05	A subqueue suffix. The **Flags** subfield MUST be 0x00. The **m_qft** member MUST be set to 0x08.

m_reserved: The integer value used for padding. The client SHOULD set this value to 0. The server MUST not use it.

(unnamed union): Based on the value of **m_qft**.

m_gPublicID: A **GUID** (as specified in [MS-DTYP] section 2.3.4) of a public queue. Selected when **m_qft** is set to 0x01.

m_oPrivateID: An OBJECTID of a private queue; members MUST be used as specified in OBJECTID. Selected when **m_qft** is set to 0x02.

m_pDirectID: A direct format name (as specified in section 2.1.2) with the "DIRECT=" prefix removed. It is selected when **m_qft** is set to 0x03.

m_gMachineID: The **GUID** (as specified in [MS-DTYP] section 2.3.4) of a machine. It is selected when **m_qft** is set to 0x04.

m_GConnectorID: The **GUID** (as specified in [MS-DTYP] section 2.3.4) of a connector queue. It is selected when **m_qft** is set to 0x05.

m_DlID: The identifier of a distribution list. It is selected when **m_qft** is set to 0x06.

m_MulticastID: A **MULTICAST_ID** (section 2.2.10) which specifies a multicast address and port. It is selected when **m_qft** is set to 0x07.

m_pDirectSubqueueID: The identifier of a subqueue. Selected when **m_qft** is set to 0x08.

The value MUST conform to the ABNF for DirectName and contain the optional <Subqueue> element, as specified in section 2.1.

The full QUEUE_FORMAT IDL is specified in [MS-MQMR] Appendix A (section 6).

2.2.8 OBJECTID

The OBJECTID structure is used to uniquely distinguish objects of the same type within the message queuing system. The structure has two identifiers: a group identifier and an object identifier.

```
typedef struct _OBJECTID {
```

```
  GUID Lineage;
  DWORD Uniquifier;
} OBJECTID;
```

Lineage: A GUID (as specified in [MS-DTYP] section 2.3.4) value that identifies the group to which an object belongs. A group is a protocol-specific concept. For instance, it can be the identifier of the object owner, or it can be the identifier of the source where the objects originate.

Uniquifier: A DWORD value that identifies the object within the group.

2.2.9 DL_ID

The DL_ID structure defines a distribution list queue identifier.

```
typedef struct _DL_ID {
  GUID m_DlGuid;
  [string] wchar_t* m_pwzDomain;
} DL_ID;
```

m_DlGuid: The GUID (as specified in [MS-DTYP] section 2.3.4) of the distribution list queue.

m_pwzDomain: The Active Directory domain of the distribution list queue. This field MUST be a null-terminated Unicode string.

2.2.10 MULTICAST_ID

The MULTICAST_ID structure defines a multicast queue identifier.

```
typedef struct _MULTICAST_ID {
  ULONG m_address;
  ULONG m_port;
} MULTICAST_ID;
```

m_address: The IP address of the queue.

m_port: The port to which the queue is attached.

2.2.11 QUEUE_SUFFIX_TYPE

The QUEUE_SUFFIX_TYPE enumeration defines which type of queue object is represented by the QUEUE_FORMAT (section 2.2.7) structure. This suffix refers to the portion of the queue path separated from the queue name by a semicolon, as specified in Path Names (section 2.1.1).

```
typedef  enum
{
  QUEUE_SUFFIX_TYPE_NONE = 0,
  QUEUE_SUFFIX_TYPE_JOURNAL = 1,
  QUEUE_SUFFIX_TYPE_DEADLETTER = 2,
  QUEUE_SUFFIX_TYPE_DEADXACT = 3,
  QUEUE_SUFFIX_TYPE_XACTONLY = 4,
  QUEUE_SUFFIX_TYPE_SUBQUEUE = 5
} QUEUE_SUFFIX_TYPE;
```

QUEUE_SUFFIX_TYPE_NONE: There is no suffix.

QUEUE_SUFFIX_TYPE_JOURNAL: Refers to the queue journal of the queue identified by the unnamed union in the QUEUE_FORMAT (section 2.2.7) structure.

QUEUE_SUFFIX_TYPE_DEADLETTER: Refers to the nontransacted dead-letter queue of the computer identified by the union in the QUEUE_FORMAT (section 2.2.7) structure.

QUEUE_SUFFIX_TYPE_DEADXACT: Refers to the transacted dead-letter queue of the computer identified by the union in the QUEUE_FORMAT (section 2.2.7) structure.

QUEUE_SUFFIX_TYPE_XACTONLY: Refers to the transacted connector queue of the connector identified by the union in the QUEUE_FORMAT (section 2.2.7) structure.

QUEUE_SUFFIX_TYPE_SUBQUEUE: Refers to the subqueue that is the direct name identified by the union in the QUEUE_FORMAT (section 2.2.7) structure.

2.2.12 PROPVARIANT Type Constants

The following values are used in the discriminant field, **vt**, of the PROPVARIANT (section 2.2.13) type.

The PROPVARIANT (section 2.2.13) type constants are defined in the VARENUM enumeration, as follows:

```
typedef enum
{
    VT_EMPTY           = 0,
    VT_NULL            = 1,
    VT_I2              = 2,
    VT_I4              = 3,
    VT_BOOL            = 11,
    VT_VARIANT         = 12,
    VT_I1              = 16,
    VT_UI1             = 17,
    VT_UI2             = 18,
    VT_UI4             = 19,
    VT_I8              = 20,
    VT_UI8             = 21,
    VT_LPWSTR          = 31,
    VT_BLOB            = 65,
    VT_CLSID           = 72,

    VT_VECTOR          = 0x1000,
} VARENUM;
```

VT_EMPTY: (0x0000): The type of the contained field is undefined. When this flag is specified, the PROPVARIANT (section 2.2.13) MUST NOT contain a data field.

VT_NULL: (0x0001): The type of the contained field is NULL. When this flag is specified, the PROPVARIANT (section 2.2.13) MUST NOT contain a data field.

VT_I2: (0x0002): The type of the contained field MUST be a 2-byte signed integer.

VT_I4: (0x0003): The type of the contained field MUST be a 4-byte signed integer.

VT_BOOL: (0x000B): The type of the contained field MUST be VARIANT_BOOL (section 2.2.14).

VT_VARIANT: (0x000C): The type of the contained field MUST be CAPROPVARIANT (section 2.2.16.8). It MUST appear with the bit flag VT_VECTOR.

VT_I1: (0x0010): The type of the contained field MUST be a 1-byte integer.

VT_UI1: (0x0011): The type of the contained field MUST be a 1-byte unsigned integer.

VT_UI2: (0x0012): The type of the contained field MUST be a 2-byte unsigned integer.

VT_UI4: (0x0013): The type of the contained field MUST be a 4-byte unsigned integer.

VT_I8: (0x0014): The type of the contained field MUST be an 8-byte signed integer.

VT_UI8: (0x0015): The type of the contained field MUST be an 8-byte unsigned integer.

VT_LPWSTR: (0x001F): The type of the contained field MUST be an **LPWSTR** (as specified in [MS-DTYP] section 2.2.36), a null-terminated Unicode string.

VT_BLOB: (0x0041): The type of the contained field MUST be a binary large object (BLOB) (section 2.2.15).

VT_CLSID: (0x0048): The type of the contained field MUST be a pointer to a **GUID** (as specified in [MS-DTYP] section 2.3.4) value.

VT_VECTOR: (0x1000): The type of the contained field MUST be combined with other values by using the bitwise OR operation to indicate a counted field. The type of the contained field MUST be a COUNTEDARRAY (section 2.2.16).

2.2.12.1 VARTYPE

The VARTYPE holds VARENUM (section 2.2.12) enumerated values.

This type is declared as follows:

```
typedef unsigned short VARTYPE;
```

2.2.13 PROPVARIANT

The PROPVARIANT (section 2.2.13.2) is a container for a union that can hold many types of data.

2.2.13.1 tag_inner_PROPVARIANT

```
typedef struct _tag_inner_PROPVARIANT {
  VARTYPE vt;
  UCHAR wReserved1;
  UCHAR wReserved2;
  ULONG wReserved3;
  [switch_is(vt)] union {
    [case(VT_EMPTY, VT_NULL)]
      ;
    [case(VT_I1)]
      CHAR cVal;
    [case(VT_UI1)]
      UCHAR bVal;
    [case(VT_I2)]
      SHORT iVal;
    [case(VT_UI2)]
      USHORT uiVal;
    [case(VT_I4)]
      LONG lVal;
    [case(VT_UI4)]
      ULONG ulVal;
    [case(VT_I8)]
      LARGE_INTEGER hVal;
    [case(VT_UI8)]
      ULARGE_INTEGER uhVal;
    [case(VT_BOOL)]
      VARIANT_BOOL boolVal;
```

```
    [case(VT_CLSID)]
      GUID* puuid;
    [case(VT_BLOB)]
      BLOB blob;
    [case(VT_LPWSTR)]
      [string] wchar_t* pwszVal;
    [case(VT_VECTOR|VT_UI1)]
      CAUB caub;
    [case(VT_VECTOR|VT_UI2)]
      CAUI caui;
    [case(VT_VECTOR|VT_I4)]
      CAL cal;
    [case(VT_VECTOR|VT_UI4)]
      CAUL caul;
    [case(VT_VECTOR|VT_UI8)]
      CAUH cauh;
    [case(VT_VECTOR|VT_CLSID)]
      CACLSID cauuid;
    [case(VT_VECTOR|VT_LPWSTR)]
      CALPWSTR calpwstr;
    [case(VT_VECTOR|VT_VARIANT)]
      CAPROPVARIANT capropvar;
  } _varUnion;
} tag_inner_PROPVARIANT;
```

vt: MUST be set to one of the values as specified in section 2.2.12.

wReserved1: MAY be set to 0x00 and MUST be ignored by the recipient.

wReserved2: MAY be set to 0x00 and MUST be ignored by the recipient.

wReserved3: MAY be set to 0x00000000 and MUST be ignored by the recipient.

_varUnion: MUST contain an instance of the type according to the value in the **vt** field.

2.2.13.2 PROPVARIANT

The following is the type definition for PROPVARIANT.

This type is declared as follows:

```
typedef tag_inner_PROPVARIANT PROPVARIANT;
```

2.2.14 VARIANT_BOOL

The VARIANT_BOOL type specifies Boolean values.

The values MUST be defined as follows.

Name/value	Value	Description
VARIANT_TRUE	0xFFFF	MUST indicate a Boolean value of TRUE.
VARIANT_FALSE	0x0	MUST indicate a Boolean value of FALSE.

This type is declared as follows:

```
typedef short VARIANT_BOOL;
```

2.2.15 BLOB

The BLOB structure defines a counted array of unsigned characters.

```
typedef struct tagBLOB {
  unsigned long cbSize;
  [size_is(cbSize)] unsigned char* pBlobData;
} BLOB;
```

cbSize: A 32-bit unsigned integer that specifies the size of the array of unsigned characters pointed to by **pBlobData**.

pBlobData: An array of 8-bit unsigned characters.

2.2.16 COUNTEDARRAY

A COUNTEDARRAY specifies a counted array of types.

2.2.16.1 CAUB

The CAUB structure defines a counted array of unsigned characters.

```
typedef struct tagCAUB {
  unsigned long cElems;
  [size_is(cElems)] unsigned char* pElems;
} CAUB;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of unsigned characters.

2.2.16.2 CAUI

The CAUI structure defines a counted array of unsigned short integers.

```
typedef struct tagCAUI {
  unsigned long cElems;
  [size_is(cElems )] unsigned short* pElems;
} CAUI;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of unsigned short integers.

2.2.16.3 CAL

The CAL structure defines a counted array of 32-bit unsigned integers.

```
typedef struct tagCAL {
  unsigned long cElems;
  [size_is(cElems )] long* pElems;
} CAL;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of 32-bit unsigned integers.

2.2.16.4 CAUL

The CAUL structure defines a counted array of 32-bit unsigned integers.

```
typedef struct tagCAUL {
  unsigned long cElems;
  [size_is(cElems )] unsigned long* pElems;
} CAUL;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of 32-bit unsigned integers.

2.2.16.5 CAUH

The CAUH structure defines a counted array of ULARGE_INTEGER (section 2.2.17) values.

```
typedef struct tagCAUH {
  unsigned long cElems;
  [size_is(cElems)] ULARGE_INTEGER* pElems;
} CAUH;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of ULARGE_INTEGER (section 2.2.17) values.

2.2.16.6 CACLSID

The CACLSID structure defines a counted array of GUID (as specified in [MS-DTYP] section 2.3.4) values.

```
typedef struct tagCACLSID {
  unsigned long cElems;
  [size_is(cElems)] GUID* pElems;
} CACLSID;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of GUID (as specified in [MS-DTYP] section 2.3.4) values.

2.2.16.7 CALPWSTR

The CALPWSTR structure defines a counted array of wchar_t* values.

```
typedef struct tagCALPWSTR {
  unsigned long cElems;
  [size_is(cElems)] [string] wchar_t** pElems;
} CALPWSTR;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of wchar_t* values.

2.2.16.8 CAPROPVARIANT

The CAPROPVARIANT structure defines a counted array of PROPVARIANT (section 2.2.13.2) values.

```
typedef struct tagCAPROPVARIANT {
  unsigned long cElems;
  [size_is(cElems )] PROPVARIANT* pElems;
} CAPROPVARIANT;
```

cElems: MUST be set to the total number of elements of the array.

pElems: An array of PROPVARIANT (section 2.2.13.2) values.

2.2.17 ULARGE_INTEGER

The ULARGE_INTEGER structure defines a large integer.

```
typedef struct _ULARGE_INTEGER {
  ULONGLONG QuadPart;
} ULARGE_INTEGER;
```

QuadPart: A **ULONGLONG** (as specified in [MS-DTYP] section 2.2.55) value.

2.2.18 Common Packet Syntax

Multiple MSMQ protocols share the packet syntax that is defined in the following sections.

The protocols that consume this packet syntax use little-endian byte order.

2.2.18.1 Packet Data Types

The following data types are used for describing the common packet syntax.

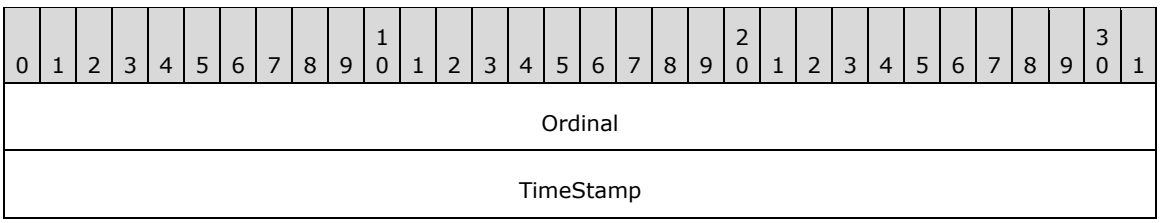
2.2.18.1.1 GUID

This specification uses the globally unique identifier data type (the **GUID** data type), as specified in [MS-DTYP] section 2.3.4.

2.2.18.1.2 TxSequenceID

A TxSequenceID is a 64-bit value that identifies a sequence of transactional messages originated from a queue manager. This structure contains two monotonically increasing numeric values.

When comparing ADM elements of type TxSequenceID, the structure is treated as a 64-bit unsigned integer with **Ordinal** being the low-order bytes and **Timestamp** being the high-order bytes.

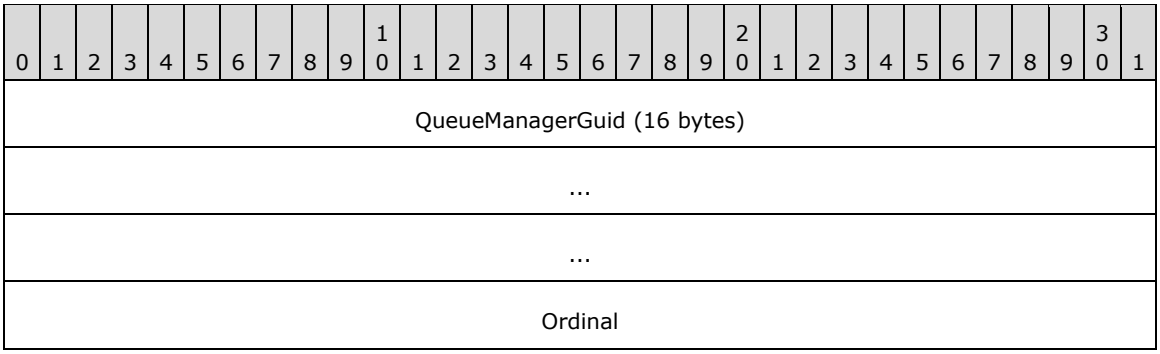


Ordinal (4 bytes): A 32-bit unsigned integer field. This field has a valid range from 0x00000000 to 0xFFFFFFFF.

TimeStamp (4 bytes): A 32-bit unsigned integer field. This field has a valid range from 0x00000000 to 0xFFFFFFFF.

2.2.18.1.3 MessageIdentifier

A MessageIdentifier is a 20-byte identifier that uniquely identifies a message from a queue manager.

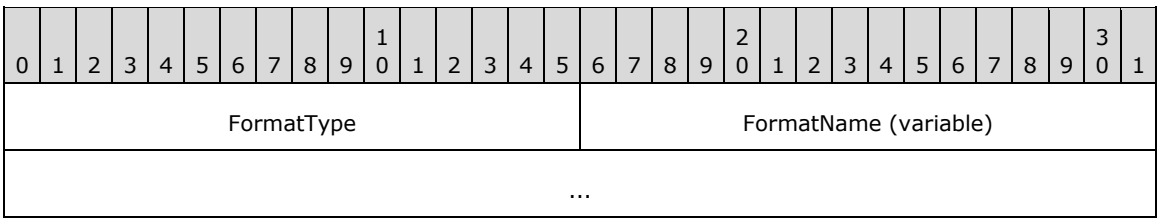


QueueManagerGuid (16 bytes): A **GUID**, as specified in [MS-DTYP] section 2.3.4, that contains the identifier of the sender queue manager. This value MUST be the same for messages originating from the same queue manager.

Ordinal (4 bytes): A 32-bit unsigned integer ordinal value that identifies the message. This value MUST be unique within messages originating from the same queue manager. This field has a valid range from 0x00000000 to 0xFFFFFFFF.

2.2.18.1.4 MQFFormatNameElement

The MQFFormatNameElement specifies a queue format name.



FormatType (2 bytes): A 16-bit unsigned short integer field specifying the queue format name type. The field MUST be set to one of the following values.

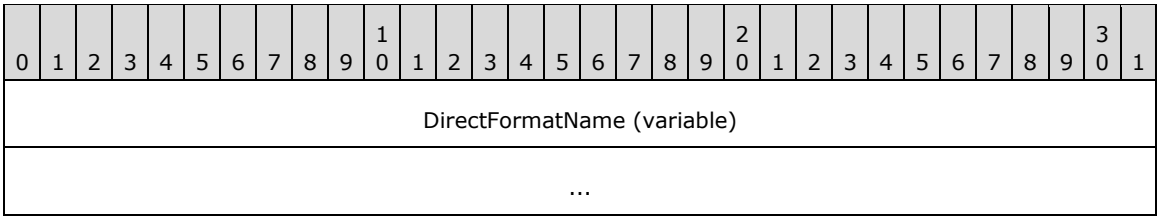
Value	Meaning
0x0001	Public Format Name
0x0002	Private Format Name
0x0003	Direct Format Name
0x0006	Distribution List Format Name

FormatName (variable): A variable-length byte array that contains a queue format name. The layout of this field depends on the value of the **FormatType** field. There are no restrictions on the value of the padding bytes.<1> The following table lists the data structures that MUST be used as the **FormatName** field for specific **FormatType** values.

FormatType Value	Data Structure
0x0001	**FormatName** contains a PublicQueueFormatName structure.
0x0002	**FormatName** contains a PrivateQueueFormatName structure.
0x0003	**FormatName** contains an MQFDirectQueueFormatName structure.
0x0006	**FormatName** contains an MQFDistributionQueueFormatName structure.

2.2.18.1.4.1 MQFDirectQueueFormatName

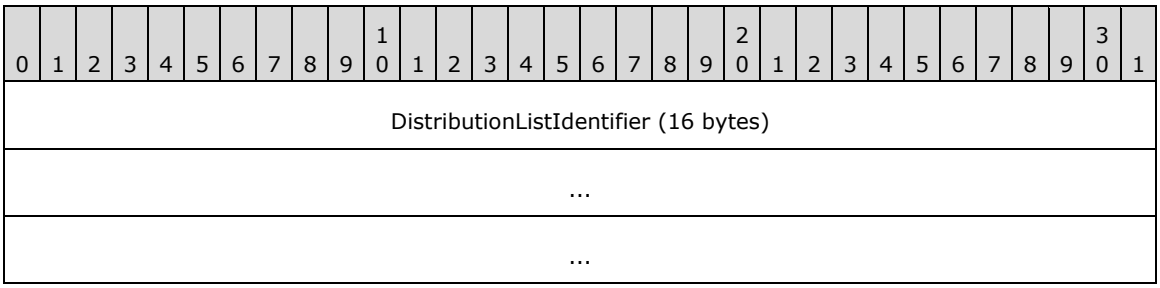
If the **FormatType** field is set to 0x0003, the layout of the **FormatName** field MUST be as follows.



DirectFormatName (variable): A null-terminated **WCHAR** ([MS-DTYP] section 2.2.60) buffer that MUST specify a queue using a direct format name. The size of the field is the **WCHAR** string's length in bytes. The start of this field is rounded up to the next 2-byte boundary.

2.2.18.1.4.2 MQFDistributionQueueFormatName

If the **FormatType** field is set to 0x0006, then the layout of the **FormatName** field MUST be as follows.



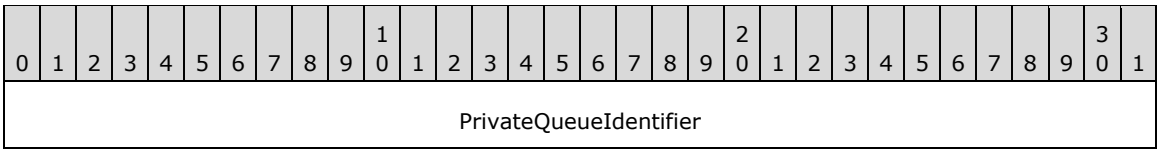
DistributionListIdentifier (16 bytes): A **GUID**, as specified in [MS-DTYP] section 2.3.4, that MUST identify a distribution list. The start of this field is rounded up to the next 4-byte boundary.

2.2.18.1.5 Queue Format Type

The Queue Format Type specifies the layout of the queue name as indicated by the **UserHeader.Flags.DQ**, **UserHeader.Flags.AQ**, and **UserHeader.Flags.RQ** fields (section 2.2.19.2) and is one of the types specified in sections 2.2.18.1.5.1 and 2.2.18.1.5.2.

2.2.18.1.5.1 PrivateQueueFormatNameId

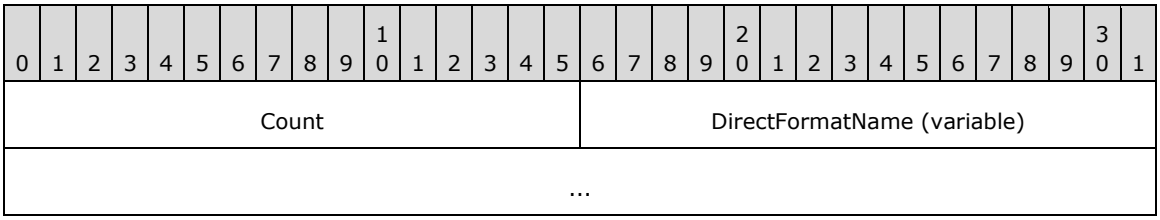
The layout of the PrivateQueueFormatNameId queue format type MUST be as follows.



PrivateQueueIdentifier (4 bytes): A 32-bit unsigned integer that identifies the private queue on the host queue manager. This value MUST be unique for private queues hosted on the same queue manager.

2.2.18.1.5.2 DirectQueueFormatName

The layout of the DirectQueueFormatName queue format type MUST be as follows.



Count (2 bytes): A 16-bit unsigned integer that specifies the length, in bytes, of the following null-terminated **WCHAR** buffer, **DirectFormatName**, up to and including the terminating null character.

DirectFormatName (variable): A null-terminated **WCHAR**, as specified in [MS-DTYP] section 2.2.60, buffer that contains a direct format name. The end of this field is padded up to the next 4-byte boundary relative to the start of the header that contains the queue format name.

The **Count** value does not include the padding bytes. There are no restrictions on the value of the padding bytes.<2>

2.2.18.1.6 Message Class Identifiers

A message class identifier is used to indicate the type of a message. In some cases the class identifier can also indicate that the message is generated as a response to an action on another message such as its receipt by a higher-layer messaging application or a delivery failure. This other message is referred to as the original message in the text that follows. The sender of the original message is referred to as the original sender in the text that follows.

The **MESSAGE_CLASS_VALUES** enumeration identifies the predefined message types.

Alternatively, the sending application can set the type of a message to a custom value by setting one or more bits of the mask 0xE1FF and by leaving the remaining bits as zero.

```
typedef  enum
{
  MQMSG_CLASS_NORMAL = 0x0000,
  MQMSG_CLASS_REPORT = 0x0001,
  MQMSG_CLASS_ACK_REACH_QUEUE = 0x0002,
  MQMSG_CLASS_ORDER_ACK = 0x00ff,
  MQMSG_CLASS_ACK_RECEIVE = 0x4000,
  MQMSG_CLASS_NACK_BAD_DST_Q = 0x8000,
  MQMSG_CLASS_NACK_DELETED = 0x8001,
  MQMSG_CLASS_NACK_REACH_QUEUE_TIMEOUT = 0x8002,
  MQMSG_CLASS_NACK_Q_EXCEED_QUOTA = 0x8003,
  MQMSG_CLASS_NACK_ACCESS_DENIED = 0x8004,
  MQMSG_CLASS_NACK_HOP_COUNT_EXCEEDED = 0x8005,
  MQMSG_CLASS_NACK_BAD_SIGNATURE = 0x8006,
  MQMSG_CLASS_NACK_BAD_ENCRYPTION = 0x8007,
```

```
  MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_Q = 0x8009,
  MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_MSG = 0x800A,
  MQMSG_CLASS_NACK_UNSUPPORTED_CRYPTO_PROVIDER = 0x800B,
  MQMSG_CLASS_NACK_Q_DELETED = 0xC000,
  MQMSG_CLASS_NACK_Q_PURGED = 0xC001,
  MQMSG_CLASS_NACK_RECEIVE_TIMEOUT = 0xC002,
  MQMSG_CLASS_NACK_RECEIVE_REJECTED = 0xC004
} MESSAGE_CLASS_VALUES;
```

MQMSG_CLASS_NORMAL: Indicates the original message sent on behalf of a higher-layer messaging application.

MQMSG_CLASS_REPORT: Indicates a report message used to track delivery of sent messages. For more information, see [MS-MQQB] section 3.1.5.8.9.

MQMSG_CLASS_ACK_REACH_QUEUE: The class is used by administration acknowledgment messages.

Indicates that the original message was delivered to its destination queue.

MQMSG_CLASS_ORDER_ACK: The class is used to acknowledge in-order receipt of an original transactional message. This acknowledgment MUST be sent from the final destination queue manager to the original sender.

MQMSG_CLASS_ACK_RECEIVE: The class is used by administration acknowledgment messages.

Indicates that the original message was retrieved by a receiving application from the destination queue.

MQMSG_CLASS_NACK_BAD_DST_Q: Indicates that the destination queue is not available to the original sender.

MQMSG_CLASS_NACK_DELETED: Indicates that the original message was deleted by an administrative action before reaching the destination queue.

MQMSG_CLASS_NACK_REACH_QUEUE_TIMEOUT: Indicates that the original message did not reach the destination queue. This message can be generated by expiration of either the UserMessage.UserHeader.TimeToBeReceived time or UserMessage.BaseHeader.TimeToReachQueue time before the original message reaches the destination queue.

MQMSG_CLASS_NACK_Q_EXCEED_QUOTA: Indicates that the original message was rejected by the destination queue manager because the destination queue exceeded Quota.

MQMSG_CLASS_NACK_ACCESS_DENIED: Indicates that the access rights for placing the original message in the destination queue were not allowed for the sender.

MQMSG_CLASS_NACK_HOP_COUNT_EXCEEDED: Indicates that the original message was rejected because it exceeded the maximum routing hop count.

MQMSG_CLASS_NACK_BAD_SIGNATURE: Indicates that the digital signature attached to the original message is not valid.

MQMSG_CLASS_NACK_BAD_ENCRYPTION: Indicates that the destination queue manager could not decrypt the original message.

MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_Q: Indicates that the original transactional message was sent to a nontransactional queue.

MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_MSG: Indicates that the original nontransactional message was sent to a transactional queue.

MQMSG_CLASS_NACK_UNSUPPORTED_CRYPTO_PROVIDER: Indicates that the encryption provider requested in the original message is not supported by the destination.

MQMSG_CLASS_NACK_Q_DELETED: Indicates that the queue was deleted before the original message could be read from the queue.

MQMSG_CLASS_NACK_Q_PURGED: Indicates that the queue was purged and the original message no longer exists.

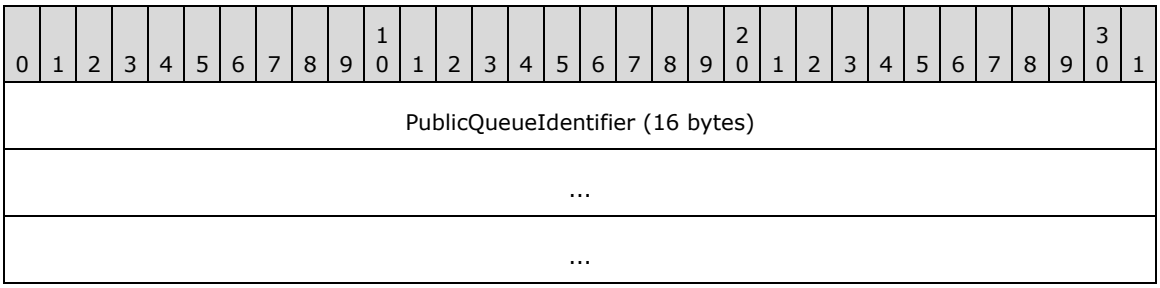
MQMSG_CLASS_NACK_RECEIVE_TIMEOUT: Indicates that the original message was placed in the destination queue but was not retrieved from the queue before its time-to-be-received timer expired.

MQMSG_CLASS_NACK_RECEIVE_REJECTED: Indicates that the message was rejected by a receiving application.

2.2.18.1.7 Common Queue Formats

2.2.18.1.7.1 PublicQueueFormatName

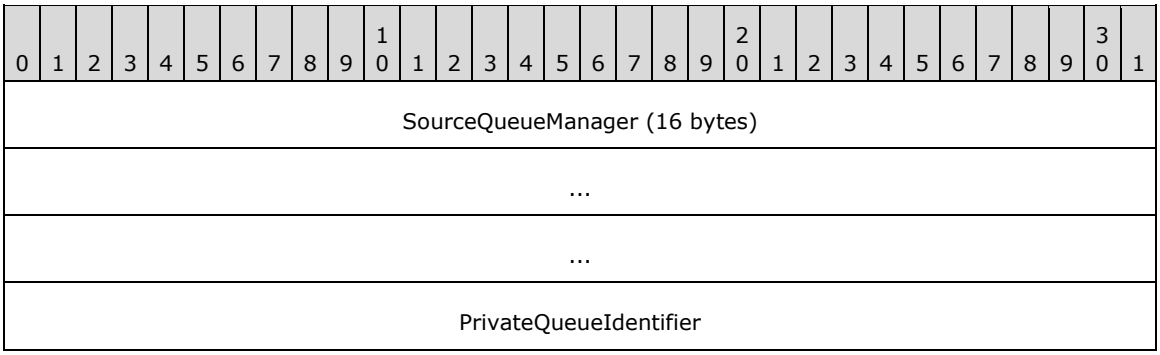
The layout of a public queue format name MUST be as follows.



PublicQueueIdentifier (16 bytes): A **GUID**, as specified in [MS-DTYP] section 2.3.4, that MUST be set to the identifier of the public queue.

2.2.18.1.7.2 PrivateQueueFormatName

The layout of a private queue format name MUST be as follows.



SourceQueueManager (16 bytes): A **GUID**, as specified in [MS-DTYP] section 2.3.4, that MUST specify the queue manager that hosts the private queue. The start of this field is rounded up to the next 4-byte boundary.

PrivateQueueIdentifier (4 bytes): A 32-bit unsigned integer that MUST identify the private queue on the source queue manager. This value MUST be unique for private queues hosted on the same queue manager.

2.2.18.1.8 XACTUOW

An XACTUOW is a structure that serves as a unique identifier for a transactional unit of work. An XACTUOW contains 16 unsigned single-byte characters representing a GUID ([MS-DTYP] section 2.3.4) and is defined as follows:

```
typedef struct {
  unsigned char rgb[16];
} XACTUOW;
```

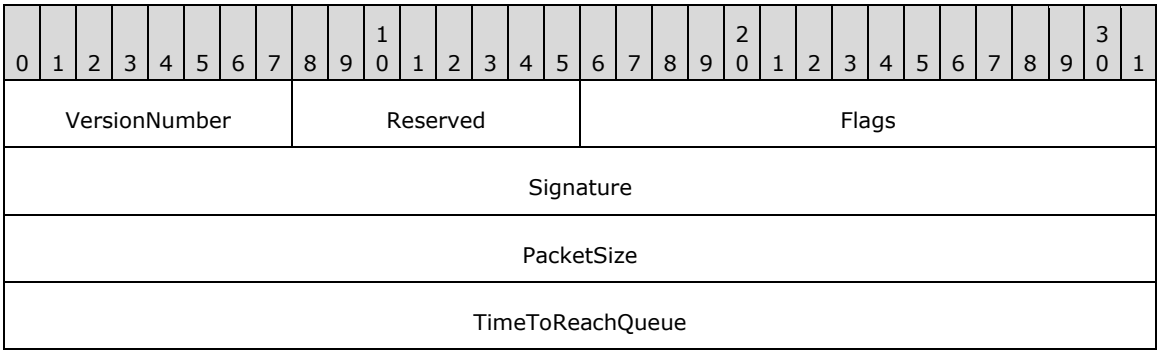
rgb: An array of unsigned single-byte characters that contains a globally unique identifier (GUID).

2.2.19 Common Headers

This section contains headers that are common to multiple packets.

2.2.19.1 BaseHeader

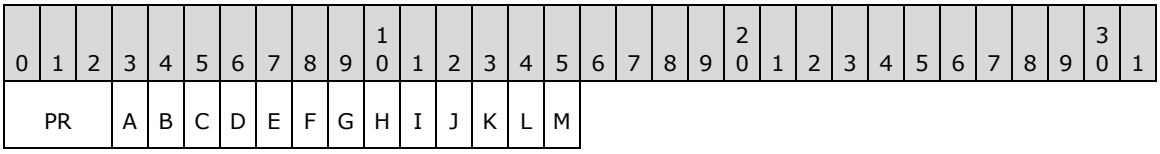
The BaseHeader is the first field of each packet described in this section. The BaseHeader contains information to identify and manage protocol packets.



VersionNumber (1 byte): An 8-bit unsigned integer that is the version of the packet format. This field MUST be set to the value 0x10.

Reserved (1 byte): Reserved for future use. This field can be set to any arbitrary value when sent and MUST be ignored on receipt.

Flags (2 bytes): A 16-bit unsigned short integer containing a set of options that provides additional information about the packet. Any combination of these values is acceptable unless otherwise noted in the following table.



PR (3 bits): Specifies the priority of the message in the packet. This field has a valid range from 0x0 to 0x7, with 0x7 being the highest priority. The default is 0x3. A message with a higher priority MUST be placed closer to the front of the queue. This field MUST be set to a value of 0x0 if the packet contains a transactional message. For more details, see **UserHeader.Flags.TH** in section 2.2.19.2.

A - IN (1 bit): Indicates that the message within the packet is internal and used by message transfer protocols for connection establishment and session acknowledgements. This field

MUST be set if the packet is an EstablishConnection Packet as defined in [MS-MQQB] section 2.2.3, a ConnectionParameters Packet as defined in [MS-MQQB] section 2.2.2, or a SessionAck Packet as defined in [MS-MQQB] section 2.2.6. This field MUST NOT be set if the packet is a OrderAck Packet as defined in [MS-MQQB] sections 2.2.4 or a FinalAck Packet as defined in [MS-MQQB] section 2.2.5.

B - SH (1 bit): Specifies if a SessionHeader (section 2.2.20.4) is present in the packet. If set, the packet MUST contain a SessionHeader.

C - DH (1 bit): Specifies if a DebugHeader (section 2.2.20.8) is present in the packet. This field MUST NOT be set if the BaseHeader is part of a packet other than a UserMessage Packet (section 2.2.20). If and only if set, MUST the packet include a DebugHeader.

D - X9 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

E - X8 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

F - TR (1 bit): Specifies whether message tracing is enabled for this packet. This field MUST be set if message tracing is required for this message. If this field is set, the **DH** field MUST also be set.

G - X6 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

H - X5 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

I - X4 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

J - X3 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

K - X2 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

L - X1 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

M - X0 (1 bit): Reserved. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

Signature (4 bytes): A 32-bit unsigned integer that is the packet signature value. This field MUST be set to 0x524F494C.

PacketSize (4 bytes): A 32-bit unsigned integer that indicates the packet size. This field MUST be set to the size, in bytes, of the entire packet including the base header and any padding bytes used to align the various message headers on 4-byte boundaries, but MUST NOT include the SessionHeader size when the SessionHeader is present. This field has a maximum value of 0x00400000.

TimeToReachQueue (4 bytes): A 32-bit unsigned integer that indicates the length of time, in seconds, that a UserMessage Packet has to reach its destination queue manager. This field has a valid range from 0x00000000 to 0xFFFFFFFF. The value 0xFFFFFFFF indicates an infinite time.

When a UserMessage Packet is sent or received, this value MUST be evaluated against the current system time and the **UserMessage.UserHeader.SentTime** field. If CURRENT_TIME - **UserMessage.UserHeader.SentTime** is greater than the value of this field, then the UserMessage Packet has expired and MUST be deleted by a sender and ignored by a receiver.

When the BaseHeader is not part of a UserMessage Packet, this value MUST be set to 0xFFFFFFFF.

For the purpose of this section, CURRENT_TIME is defined as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 Coordinated Universal Time (UTC).

2.2.19.2 UserHeader

The UserHeader contains source and destination information for the message in a UserMessage Packet.

SourceQueueManager (16 bytes): A **GUID**, as specified in [MS-DTYP] section 2.3.4, that MUST identify the original sender of the message.

QueueManagerAddress (16 bytes): A **GUID**, as specified in [MS-DTYP] section 2.3.4, that MUST identify the destination queue manager.

If the message is sent to a public or private queue, this field MUST be set to the **GUID** of the destination queue manager. If the message is sent to a queue that uses a direct format name, then this field MUST contain a NULL GUID.

TimeToBeReceived (4 bytes): A 32-bit unsigned integer that indicates the length of time, in seconds, that the message in the packet has before it expires. This field has a valid range from 0x00000000 to 0xFFFFFFFF. The value 0xFFFFFFFF indicates an infinite time.

This time is measured from when the sending protocol receives the message. If the value is exceeded, the message MUST be removed from the destination queue. For more details about message expiration see [MS-MQQB] section 3.1.5.8.5.

SentTime (4 bytes): A 32-bit unsigned integer that MUST be set to the time when the packet was sent. This value represents the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

MessageID (4 bytes): A 32-bit unsigned integer that is the message identifier specified by the queue manager. The queue manager MUST generate a unique identifier for each message it sends. For more details, see the **MessageIdOrdinal** value in [MS-MQQB] section 3.1.1.3.

Flags (4 bytes): A 32-bit unsigned integer that contains a set of options that provide additional information about the packet. Any combination of these values is acceptable unless otherwise noted below.

RC (5 bits): The number of routing servers that have processed the UserMessage Packet. The values in this field MUST be in the range from 0x00 to 0x1D. For more details, see [MS-MQQB] section 3.1.5.8.2.

DM (2 bits): The delivery mode of the packet. The field MUST be set to one of the following values.

Value	Meaning
0x0	Express messaging. Express messages MUST NOT be required to be recoverable after the queue manager restarts.
0x1	Recoverable messaging (including transactional). Recoverable messages MUST be recovered after queue manager restarts.

Note a transactional message is a recoverable message that has UserHeader.Flags.TH set to 0x1.

A - X1 (1 bit): Reserved bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

B - JN (1 bit): Specifies if negative source journaling is enabled.<3> If set, the protocol SHOULD log a record locally in the event of message delivery failure.<4>

C - JP (1 bit): Specifies if positive source journaling is enabled. If set, the protocol SHOULD log a record locally if the message is successfully delivered.<5>

DQ (3 bits): Type of destination queue in **UserHeader.DestinationQueue**. The field MUST be set to 0x0, 0x3, 0x5, or 0x7. The value in this field determines the layout of the destination queue name in the **UserHeader.DestinationQueue** field.

AQ (3 bits): Type of administration queue in **UserHeader.AdminQueue**. The field MUST be set to 0x0, 0x2, 0x3, 0x5, 0x6 or 0x7. The value in this field determines the layout of the administration queue name in the **UserHeader.AdminQueue** field.

RQ (3 bits): Type of response queue in **UserHeader.ResponseQueue**. The field MUST be set to 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, or 0x7. The value in this field determines the layout of the response queue name in the **UserHeader.ResponseQueue** field.

D - SH (1 bit): Specifies if a SecurityHeader is present in the UserMessage Packet. If set, the packet MUST contain a SecurityHeader; otherwise, it MUST NOT.

E - TH (1 bit): Specifies if a TransactionHeader is present in the UserMessage Packet. If set, the packet MUST contain a TransactionHeader; otherwise, it MUST NOT. If this flag is set the **DM** flag MUST NOT be 0x0.

F - MP (1 bit): Specifies if a MessagePropertiesHeader is present in the UserMessage Packet. This flag MUST always be set.

G - CQ (1 bit): Specifies if the **ConnectorType** field is present in the packet. If set, the packet MUST contain a **ConnectorType** field; otherwise, it MUST NOT.

H - MQ (1 bit): Specifies if a MultiQueueFormatHeader is present in the UserMessage Packet. If set, the packet MUST contain a MultiQueueFormatHeader; otherwise, it MUST NOT.

I - X2 (1 bit): Reserved bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

J - AH (1 bit): Indicates if the packet being received was originally sent over HTTP, as specified in [MS-MQRR] section 2.2.5.1. This field MUST NOT be set when sent.

X3 (2 bits): Reserved bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

K - HH (1 bit): Specifies if a SoapHeader is present in the packet. If set, the UserMessage Packet MUST contain a SoapHeader; otherwise, it MUST NOT.

X4 (3 bits): Reserved bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

DestinationQueue (variable): The destination queue specifies the final destination of the message that is contained inside the UserMessage Packet. The queue type and data type of the destination queue name vary depending on the value specified in the **Flags.DQ** field, as described in the following table.

Flags.DQ	Queue type	Data type
0x0	None	None
0x3	Private queue on destination host	PrivateQueueFormatNameId
0x5	Public queue	PublicQueueFormatName
0x7	Direct	DirectQueueFormatName

When the **Flags.DQ** field is set to 0x03 or 0x5, the **QueueManagerAddress** field MUST be set to the **GUID** of the destination queue manager.

Any value for **Flags.DQ** other than those specified in the preceding table MUST be treated as an error by closing the session.

AdminQueue (variable): The name of the administration queue. This field specifies the response queue where administration acknowledgment messages are sent. An administration response queue MUST be specified if a MessagePropertiesHeader is included and any bits are set in the **MessagePropertiesHeader.Flags** field; otherwise, this field MUST NOT be specified. Details about administration acknowledgments are as specified in [MS-MQQB] sections 1.3.5.2 and 3.1.5.8.10. The queue type and data type of the administration queue name vary depending on the value specified in the **Flags.AQ** field, as described in the following table. This field MUST be present when the **Flags.AQ** field is set to 0x2, 0x3, 0x5, 0x6, or 0x7. This field MUST NOT be present when the **Flags.AQ** field is set to 0x0.

Flags.AQ	Queue type	Data type
0x0	None	None
0x2	Private queue on source host	PrivateQueueFormatNameId
0x3	Private queue on destination host	PrivateQueueFormatNameId
0x5	Public queue	PublicQueueFormatName
0x6	Private queue on host other than the source or destination host	PrivateQueueFormatName
0x7	Direct	DirectQueueFormatName

Any value for the **Flags.AQ** field other than those specified in the preceding table MUST be treated as an error by closing the session.

ResponseQueue (variable): A variable-length array of bytes containing the name of the response queue. The response queue is an application-defined value that specifies a queue that a receiving application could use to send a reply message. The queue type and data type of the response queue name vary depending on the queue format type specified in the **Flags.RQ** field, as described in the following table. This field MUST be present when the **Flags.RQ** field is set to 0x1, 0x2, 0x3, 0x5, 0x6, or 0x7. This field MUST NOT be present when the **Flags.RQ** field is set to 0x0 or 0x1. When the **Flags.RQ** flag is set to 0x1, the response queue is the same as the administration queue. When the **Flags.RQ** field is set to 0x4, the **PrivateQueueIdentifier** in the queue format type MUST identify the private queue on the queue manager that hosts the administration queue.

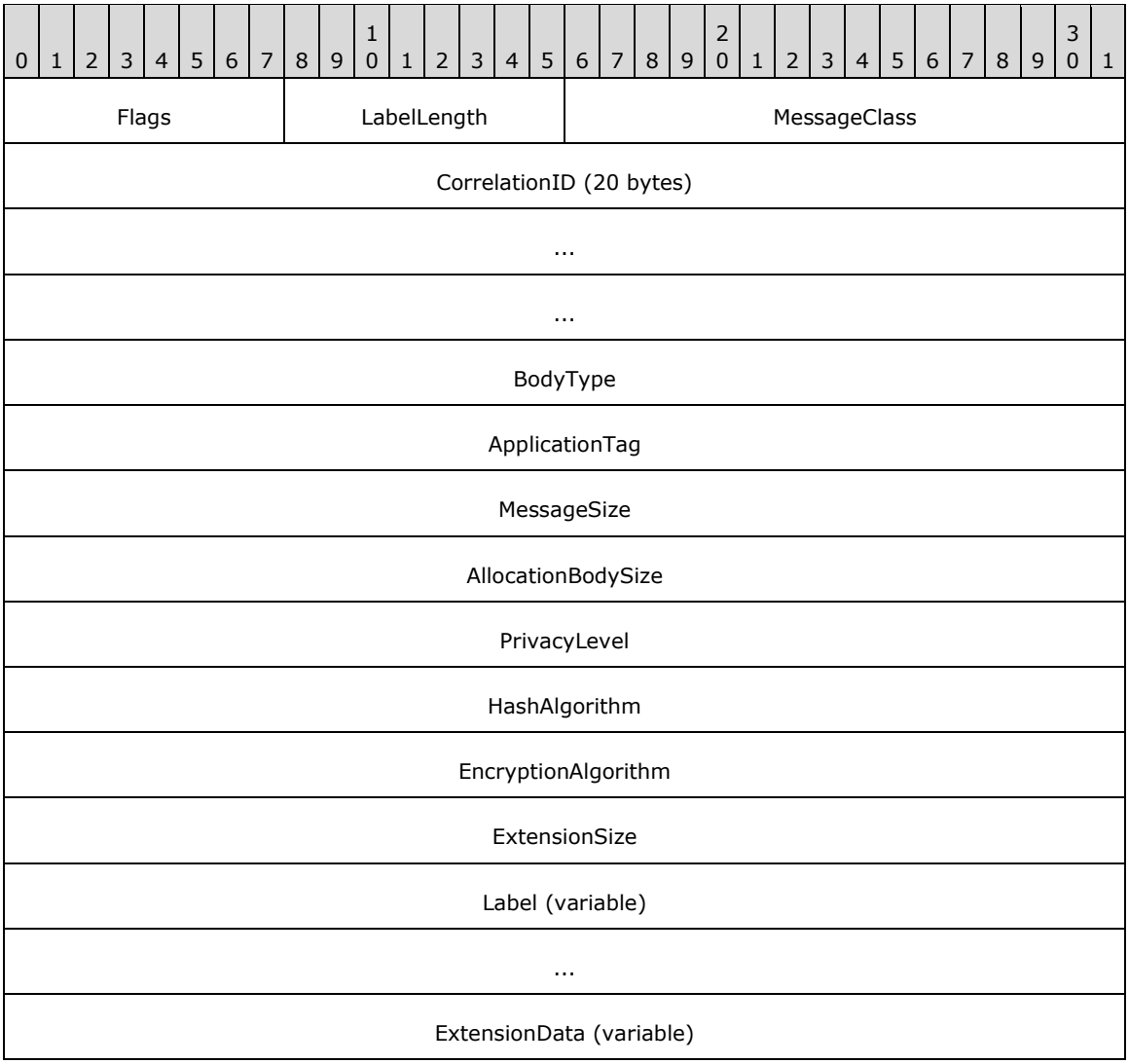
Flags.RQ	Queue type	Data type
0x0	None	None
0x1	Same as the administration queue	None
0x2	Private queue on source host	PrivateQueueFormatNameId
0x3	Private queue on destination host	PrivateQueueFormatNameId
0x4	Private queue on the same host as the administration queue	PrivateQueueFormatNameId

Flags.RQ	Queue type	Data type
0x5	Public queue	PublicQueueFormatName
0x6	Private queue on a host other than the source queue, destination queue, or administration queue host	PrivateQueueFormatName
0x7	Direct	DirectQueueFormatName

ConnectorType (16 bytes): An optional field that represents an application-defined **GUID**, as specified in [MS-DTYP] section 2.3.4. This field MUST be present if and only if the **Flags.CQ** field is set. This field is used by higher-layer messaging applications. The server MUST NOT process or interpret this field.

2.2.19.3 MessagePropertiesHeader

The MessagePropertiesHeader contains property information about a UserMessage Packet and the application-defined message payload.



...
MessageBody (variable)
...

Flags (1 byte): An 8-bit unsigned integer field that specifies administration acknowledgments. Any combination of these values is acceptable unless otherwise noted in the following table.

For more details on administration acknowledgments, see [MS-MQQB] sections 3.1.7.2.1 and 3.1.5.8.10.

0	1	2	3	4	5	6	7
PA	PR	NA	NR	X	X	X	X

Where the bits are defined as:

Value	Description
PA	If the message is delivered to the destination queue, the server MUST send a positive acknowledgment.
PR	If the message is retrieved from the destination queue by the application, the server MUST send a positive acknowledgment.
NA	If the message is not delivered to the destination queue, the server MUST send a negative acknowledgment.
NR	If the message is not retrieved from the destination queue by the application, the server MUST send a negative acknowledgment.
X	Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.
X	Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.
X	Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.
X	Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

LabelLength (1 byte): An 8-bit unsigned integer field that MUST be set to the number of elements of **WCHAR** in the **Label** field. This field has a valid range from 0x00 to 0xFA. When the value of this field is 0x00, the **Label** field MUST NOT be present after the **ExtensionSize** field. When greater than 0x00, this value MUST include the terminating null character.

MessageClass (2 bytes): A 16-bit unsigned integer that specifies the class of the message in the packet. The value MUST be set to a value of the **MESSAGE_CLASS_VALUES** enumeration specified in section 2.2.18.1.6.

CorrelationID (20 bytes): If this header appears outside an administration acknowledgement message, as specified in [MS-MQQB] section 3.1.5.8.10, then this field MUST be treated as an application-defined buffer and the server MUST not process or interpret this field.

If this header appears inside an administration acknowledgment message, as specified in [MS-MQQB] section 3.1.5.8.10, then this field MUST be set to a MessageIdentifier consisting of **UserMessage.UserHeader.MessageID** and **UserMessage.UserHeader.SourceQueueManager** of the message being acknowledged. See section 2.2.18.1.3 for details of the MessageIdentifier type.

BodyType (4 bytes): A 32-bit unsigned integer that specifies the type of data that is contained in the message body. This value MUST be set to a PROPVARIANT type constant as specified in section 2.2.12.

ApplicationTag (4 bytes): A 32-bit unsigned integer that specifies an application-defined value that can be used to organize messages and the server MUST not process or interpret this field.

MessageSize (4 bytes): A 32-bit unsigned integer that MUST be set to the size, in bytes, of the **MessageBody** field. The field MUST be set to a value between 0x00000000 and the size limit imposed by the value of **BaseHeader.PacketSize**.

AllocationBodySize (4 bytes): A 32-bit unsigned integer field that MUST be set to the size, in bytes, of the data allocated for the **MessageBody** field. This size can be larger than the actual message body size; for example, an encrypted message body might be larger than the original unencrypted message body, up to the size limit imposed by the value of **BaseHeader.PacketSize**.

PrivacyLevel (4 bytes): A 32-bit unsigned integer field that specifies the privacy level of the message in the UserMessage Packet. The privacy level determines what part of the message is encrypted. The field MUST be set to one of the following values.<6>

Value	Meaning
0x00000000	No encryption. The **MessageBody** field is sent as clear text.
0x00000001	The **MessageBody** field is encrypted using 40-bit end-to-end encryption.
0x00000003	The **MessageBody** field is encrypted using 128-bit end-to-end encryption.
0x00000005	The **MessageBody** field is encrypted using Advanced Encryption Standard (AES).

Any value not specified in the preceding table MUST be treated as an authentication failure.

HashAlgorithm (4 bytes): A 32-bit unsigned integer that specifies the hashing algorithm that is used when authenticating the message. The following table lists the allowed values for this field.

Value	Meaning
0x00008001	Specifies the MD2 hash algorithm, as specified in [RFC1319].
0x00008002	Specifies the MD4 hash algorithm, as specified in [RFC1320].
0x00008003	Specifies the MD5 hash algorithm, as specified in [RFC1321].
0x00008004	Specifies the SHA-1 hash algorithm, as specified in [RFC3110].
0x0000800C	Specifies the SHA-256 hash algorithm [FIPS180-2].
0x0000800E	Specifies the SHA-512 hash algorithm [FIPS180-2].

Any value not specified in the preceding table MUST be treated as an authentication failure when the **SecurityHeader** is present in the UserMessage and the

SecurityHeader.SecurityData.Signature field is present. The SHA-512 hash algorithm (0x0000800E) SHOULD be used to generate the message signature.<7>

EncryptionAlgorithm (4 bytes): A 32-bit unsigned integer that specifies the encryption algorithm used to encrypt the **MessageBody** field. This field MUST be set to a value in the following table.

Value	Meaning
0x00006602	Specifies the RC2 algorithm, as specified in [RFC2268].
0x00006610	Specifies the AES 256 algorithm, as specified in [FIPS197].
0x0000660E	Specifies the AES 128 algorithm, as specified in [FIPS197].
0x0000660F	Specifies the AES 192 algorithm, as specified in [FIPS197].
0x00006801	Specifies the RC4 algorithm, as specified in [RFC4757].

Any value not specified in the preceding table MUST be treated as a failed decryption error when the **SecurityHeader** is present in the UserMessage and the **SecurityHeader.EB** flag is set.

This field MUST be set according to **PrivacyLevel** as specified in the following table.

PrivacyLevel	Allowed encryption algorithms
0x00000001	RC2, RC4
0x00000003	RC2, RC4
0x00000005	AES 128, AES 192, AES 256

When specifying AES or RC2, the initialization vector must be set to zero. The Padding used with AES or RC2 is [PKCS5] padding.

ExtensionSize (4 bytes): A 32-bit unsigned integer field that MUST be set to the length, in bytes, of the application-defined **ExtensionData** field. The field MUST be set to a value between 0x00000000 and the size limit imposed by the value of **BaseHeader.PacketSize**.

Label (variable): The **Label** field is an application-defined Unicode string. This field can be used by an application to assign a short descriptive string to the message. This field is of length **LabelLength** * 2 bytes and MUST NOT be more than 500 bytes. If **LabelLength** is nonzero, this field MUST be in the format specified by the following ABNF rule.

```
label = 0*249(%x0001-FFFF) 0x0000
```

If **LabelLength** is zero then this field MUST NOT be present.

Unlike the fields preceding and including the **Label** field, the **ExtensionData** and **MessageBody** fields are not guaranteed to be on 4-byte boundaries.

ExtensionData (variable): This field is a buffer containing additional application-defined information that is associated with the message. This field is of length **ExtensionSize** bytes. If **ExtensionSize** is zero then this field MUST NOT be present.

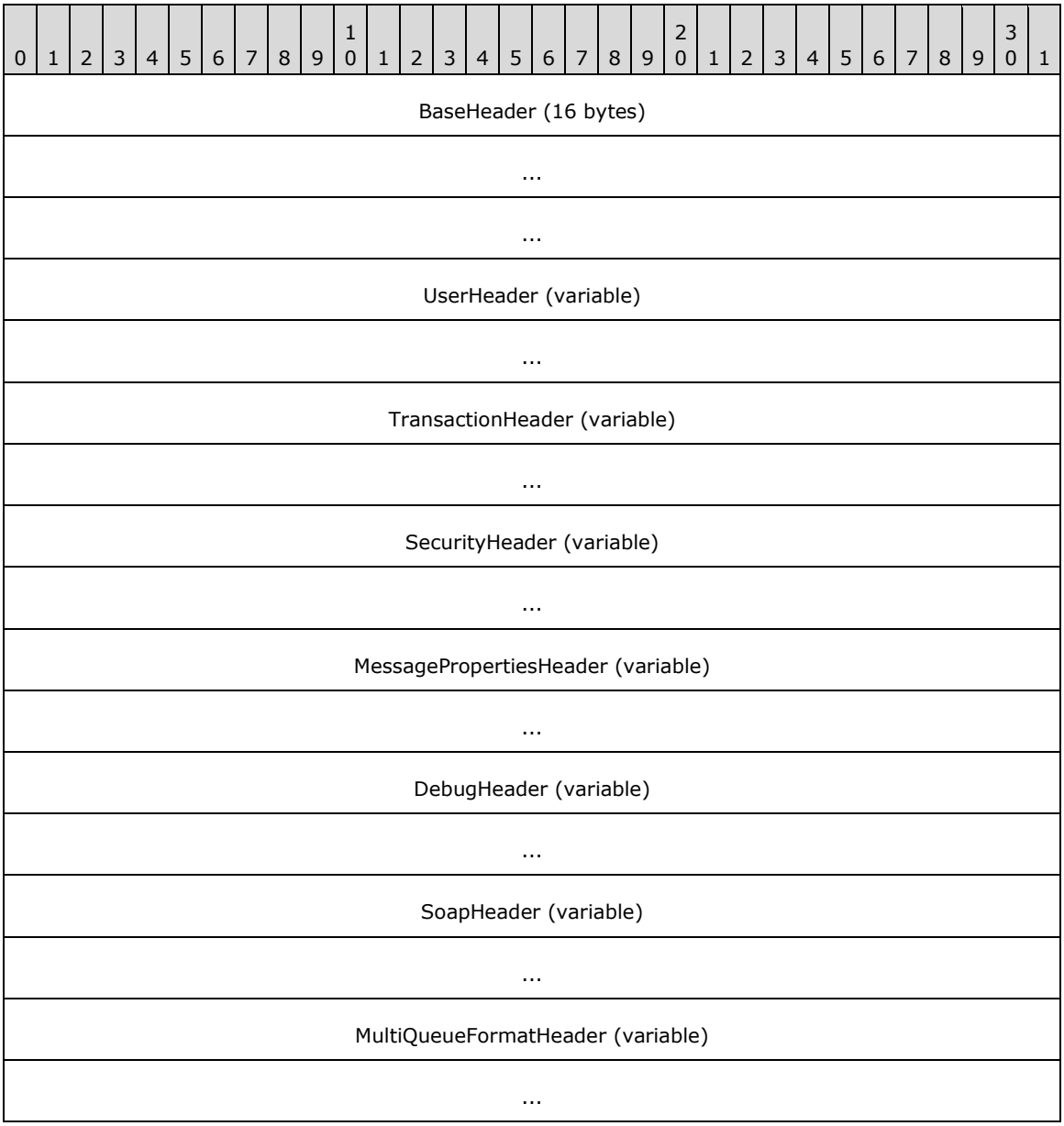
MessageBody (variable): The **MessageBody** field is a buffer containing the application-defined message payload. This field is of length **MessageSize** bytes. If **MessageSize** is zero then this field MUST NOT be present.

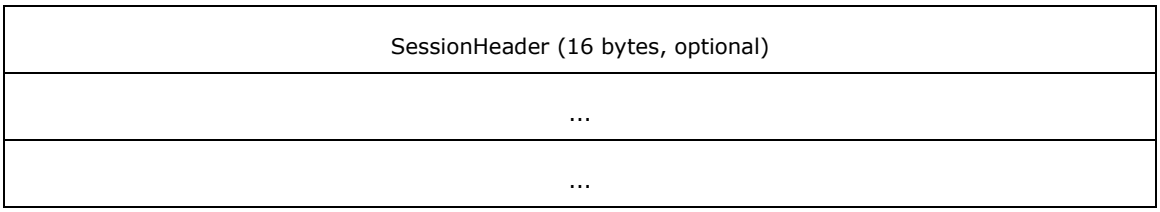
The MessagePropertiesHeader packet MUST be a multiple of 4 bytes in length and MUST append padding bytes needed to ensure this requirement. There are no restrictions on the value of the padding bytes.<8>

2.2.20 UserMessage Packet

A UserMessage Packet always contains an entire message. The UserMessage Packet is used to communicate application-defined and administration acknowledgment messages between a sender and receiver.

A UserMessage Packet contains a number of required headers and can contain additional optional headers. The required headers that MUST appear in all UserMessage Packets are: BaseHeader, UserHeader, and MessagePropertiesHeader. Optional headers include: TransactionHeader, SecurityHeader, DebugHeader, SoapHeader, MultiQueueFormatHeader, and SessionHeader.





BaseHeader (16 bytes): A BaseHeader (section 2.2.19.1) packet that contains information to identify and manage protocol packets. The **BaseHeader.Flags.IN** field MUST NOT be set.

UserHeader (variable): A UserHeader (section 2.2.19.2) packet that contains source and destination queue information.

TransactionHeader (variable): A TransactionHeader (section 2.2.20.5) packet that contains flags and sequence information for the packet. This header MUST be present when **UserHeader.Flags.TH** is set and MUST NOT be present if it is clear.

SecurityHeader (variable): A SecurityHeader (section 2.2.20.6) packet that contains security information. This header MUST be present when **UserHeader.Flags.SH** is set and MUST NOT be present if it is clear.

MessagePropertiesHeader (variable): A MessagePropertiesHeader (section 2.2.19.3) packet that contains property information about a UserMessage Packet and the application-defined message payload. This header MUST be present.

DebugHeader (variable): A DebugHeader (section 2.2.20.8) packet that specifies the queue to receive trace messages for this UserMessage Packet. This header specifies the queue where trace messages are sent. This header MUST be present if and only if **BaseHeader.Flags.DH** is set.

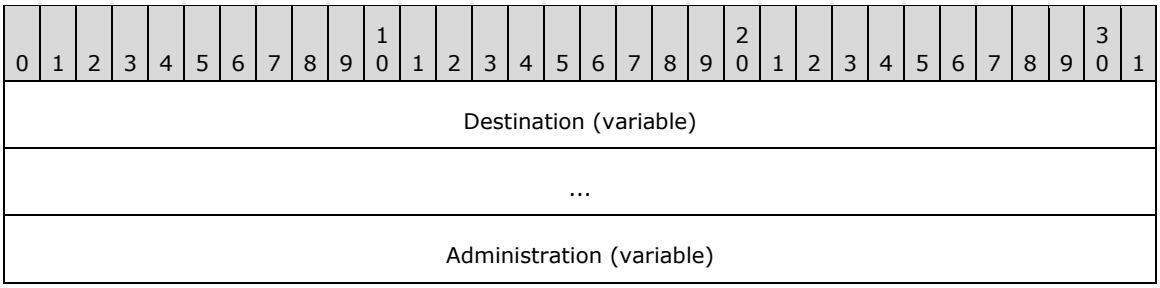
SoapHeader (variable): A SoapHeader (section 2.2.20.7) packet that contains application-defined information. This header MUST be present if and only if **UserHeader.Flags.HH** is set.

MultiQueueFormatHeader (variable): A MultiQueueFormatHeader (section 2.2.20.1) packet that is included when a message is destined for multiple queues. This header MUST be present if and only if **UserHeader.Flags.MQ** is set.

SessionHeader (16 bytes): A SessionHeader (section 2.2.20.4) packet that is used to acknowledge express and recoverable UserMessage Packets received by the message transfer protocols. This header MUST be present if and only if **BaseHeader.Flags.SH** is set.

2.2.20.1 MultiQueueFormatHeader

The optional MultiQueueFormatHeader is used when a message is destined for multiple queues.<9> When an application-layer message is sent using a multiple-element format name, this header is added to the packet to list the destinations. The sending queue manager creates a separate UserMessage Packet for each destination and specifies the packet address in the UserHeader. The information in this header provides a list of all destinations that were sent the message in addition to associated administration and response queues.



...
Response (variable)
...
Signature (variable)
...

Destination (variable): An MQFAddressHeader that specifies the name of one or more destination queues. This field MUST contain the list of all queues that were sent a copy of this UserMessage Packet. The end of this field is rounded up to the next 4-byte boundary. There are no restrictions on the value of the padding bytes.<10>

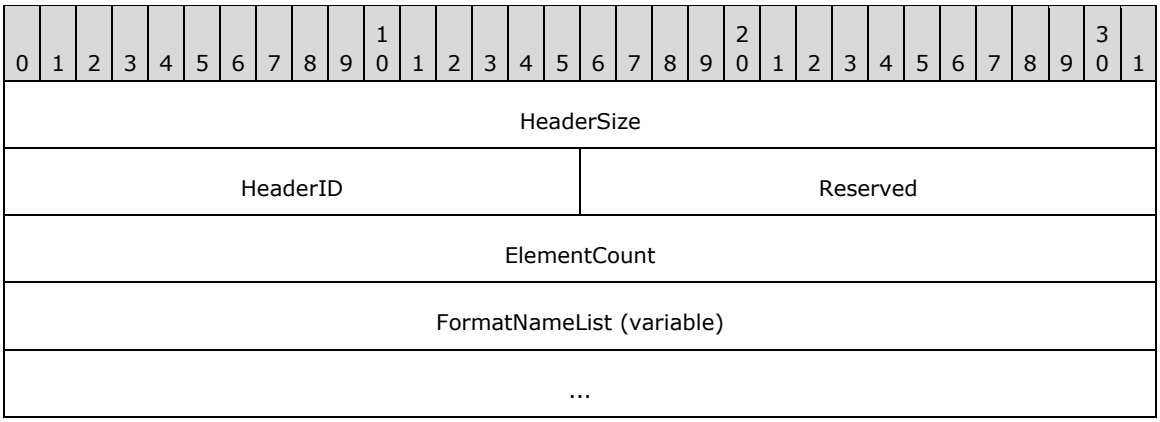
Administration (variable): An MQFAddressHeader that specifies the name of one or more administration queues that can receive positive and negative acknowledgement messages. The end of this field is rounded up to the next 4-byte boundary. There are no restrictions on the value of the padding bytes.<11>

Response (variable): An MQFAddressHeader that specifies the name of one or more response queues that can receive response messages from the receivers at the destinations. The end of this field is rounded up to the next 4-byte boundary. There are no restrictions on the value of the padding bytes.<12>

Signature (variable): An MQFSignatureHeader that specifies a signature for the packet.

2.2.20.2 MQFAddressHeader

The MQFAddressHeader is used to specify multiple destination queue format names.



HeaderSize (4 bytes): A 32-bit unsigned integer that specifies the size of the header. This value MUST contain the size, in bytes, of this header including the variable data. This field has a valid range between 0x0000000C and the size limit imposed by the value of **BaseHeader.PacketSize**.

HeaderID (2 bytes): A 16-bit unsigned integer that specifies an identifier for this header. This field MUST be set to one of the following values based on the header designation:

Value	Meaning
0x0064	Destination

Value	Meaning
0x00C8	Admin
0x012C	Response
0x015E	Signature

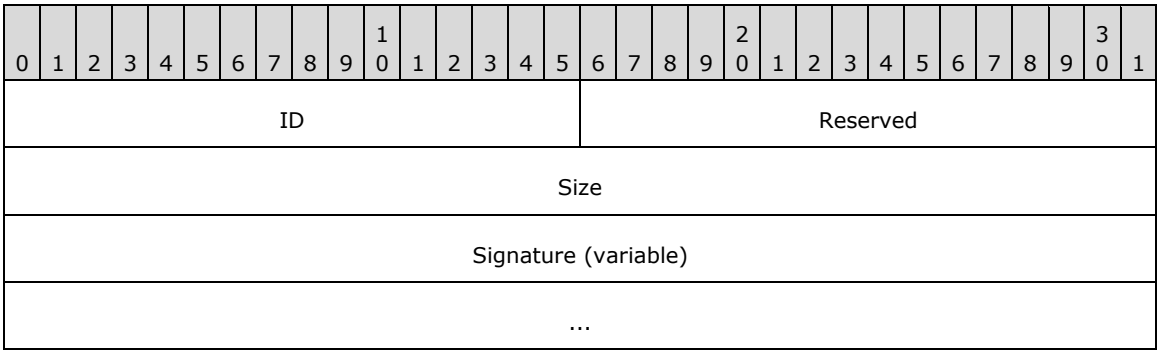
Reserved (2 bytes): A 16-bit unsigned integer field that is reserved for alignment. The sender SHOULD set this field to 0x0000, and the receiver MUST ignore it on receipt.

ElementCount (4 bytes): A 32-bit unsigned integer field that MUST be set to the number of elements in the **FormatNameList** field. This field has a valid range between 0x00000000 and the size limit imposed by the value of BaseHeader.PacketSize.

FormatNameList (variable): An MQFFormatNameElement that contains a list of queue format names. This field MUST contain a list of MQFFormatNameElement data structures. The array MUST contain the number of elements specified by the **ElementCount** field. The end of this field is rounded up to the next 4-byte boundary. Padding bytes in this field MAY be any value.<13>

2.2.20.3 MQFSignatureHeader

The MQFSignatureHeader is a signature used in the MultiQueueFormatHeader.



ID (2 bytes): A 16-bit unsigned short integer value that MUST be set to 0x015E.

Reserved (2 bytes): A 16-bit unsigned short integer field that is reserved for alignment. The sender SHOULD set this field to 0x0000, and the receiver MUST ignore it on receipt.

Size (4 bytes): A 32-bit unsigned integer field that MUST be set to the size, in **bytes**, of the **byte** array to hold the **Signature**. This field has a valid range between 0x00000000 and the size limit imposed by the value of **BaseHeader.PacketSize**.

Signature (variable): A **byte** array that contains the signature. The array MUST contain the number of elements that are specified by the **Size** field. The **Signature** MUST be calculated in the same way as the **SecurityData.Signature** as specified in section 2.2.20.6.

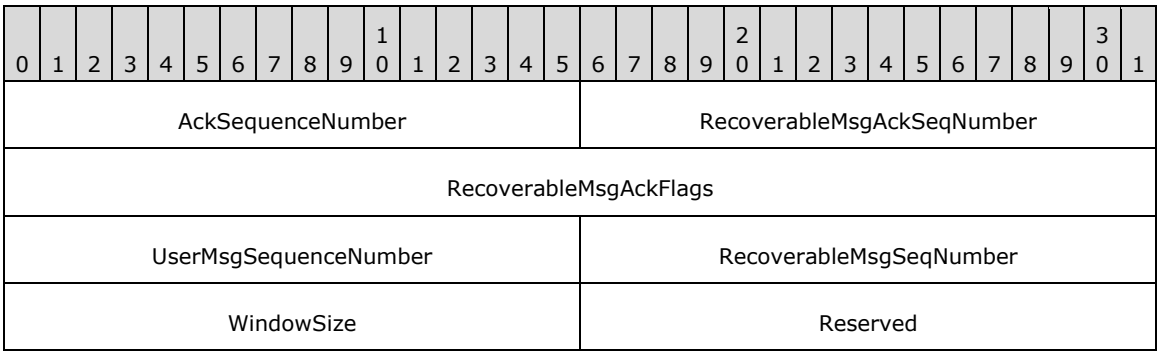
2.2.20.4 SessionHeader

The SessionHeader is used to acknowledge express and recoverable UserMessage Packets received by the message transfer protocols. This header is present in stand-alone SessionAck Packet as defined in [MS-MQQB] (section 2.2.6) and is optional in a UserMessage Packet.

This header contains a session acknowledgment. For more details, see [MS-MQQB] sections 3.1.1.7, and 3.1.1.6.1.

The set of UserMessage Packets sent on a session represent a message sequence. There is a local-to-remote and remote-to-local sequence. These message sequences exist for the lifetime of the session. The local and remote protocols MUST maintain counts of the UserMessage Packets sent and received. A message MUST be associated with a sequence number that corresponds to its position within the sequence. Sequence numbers MUST begin with 1 and MUST increment by 1 with each subsequent message. For example, the third message sent on a session has a sequence number of 3.

The protocols MUST also maintain a count of recoverable UserMessage Packets sent and associates recoverable sequence numbers with those messages. For example, the fifth recoverable message sent on a session has a sequence number of 5.



AckSequenceNumber (2 bytes): A 16-bit unsigned integer that specifies a count of messages received. This field MUST be set to the count of UserMessage Packets received on this session. This field acknowledges all messages up to and including the specified sequence number. This field has a valid range from 0x0000 to 0xFFFF, inclusive.

RecoverableMsgAckSeqNumber (2 bytes): A 16-bit unsigned integer that specifies a recoverable message sequence number. This field MUST be set to the lowest unacknowledged recoverable message sequence number that has been persisted for reliable recovery. If no recoverable messages have been received by the receiver since the last SessionHeader was sent, this field MUST be set to 0. This field has a valid range from 0x0000 to 0xFFFF, inclusive.

RecoverableMsgAckFlags (4 bytes): A 32-bit unsigned integer bit field representing messages. This bit field represents up to 32 recoverable UserMessage Packets that are being acknowledged as written to disk. Bit 0 of this field represents the UserMessage Packet whose sequence number is specified in the **RecoverableMsgAckSeqNumber** field. A given bit k of this field represents a recoverable UserMessage Packet with a sequence number of **RecoverableMsgAckSeqNumber** + k. The corresponding bit for a UserMessage Packet that has been persisted for reliable recovery MUST be set in the bit field.

UserMsgSequenceNumber (2 bytes): A 16-bit unsigned integer that is the count of messages sent. This field SHOULD<14> be set to the count of UserMessage Packets sent on this session. When the **UserMsgSequenceNumber** is not set to the count of UserMessage Packets sent on a session, the user message is sent to the destination queue, and the session is closed by the receiver. This field has a valid range from 0x0000 to 0xFFFF, inclusive.

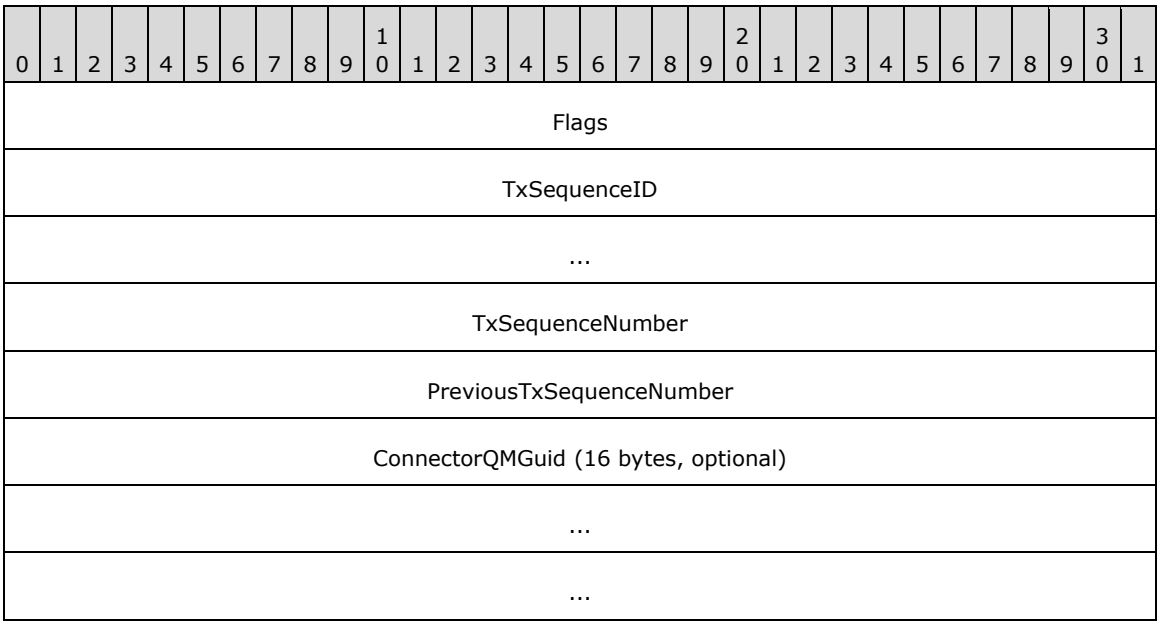
RecoverableMsgSeqNumber (2 bytes): A 16-bit unsigned integer that is the count of recoverable messages sent. This field MUST be set to the count of recoverable UserMessage Packets sent on this session. This value MUST be 0 if no recoverable UserMessage Packets have been sent. This field has a valid range from 0x0000 to 0xFFFF, inclusive.

WindowSize (2 bytes): A 16-bit unsigned integer field that specifies the acknowledgment window size. The window size controls the frequency at which the message transfer protocols send acknowledgment packets.<15> The value of this field SHOULD be set to 0x0040.<16> This field has a valid range from 0x0001 to 0xFFFF, inclusive.<17>

Reserved (2 bytes): Reserved. Can be set to any arbitrary value when sent and MUST be ignored on receipt.

2.2.20.5 TransactionHeader

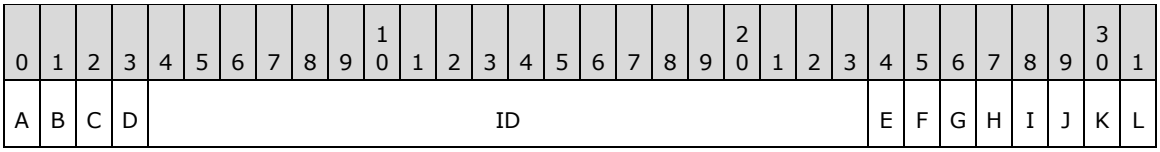
The TransactionHeader packet contains sequence information for a transactional message. The presence of this packet in a UserMessage Packet indicates that the message contained in the packet is transactional.



Flags (4 bytes): A 32-bit unsigned integer that contains a set of options that provide additional information about the packet. Any combination of these values is acceptable unless otherwise noted in the following table.

Any value not specified in the table MUST be treated as an error by closing the session.

The value SHOULD be set to a combination of the following values.<18>



A - CG (1 bit): A bit that specifies whether the **ConnectorQMGuid** field contains a connector queue manager **GUID**. If set, the **ConnectorQMGuid** field MUST contain a **GUID**.

B - FA (1 bit): A bit that specifies whether a FinalAck Packet, as defined in [MS-MQQB] section 2.2.5, is required. For more details see [MS-MQQB] section 3.1.7.2.2.

C - FM (1 bit): A bit that specifies whether the message is the first one sent within the context of a transaction. This bit MUST be set if the message is the first one in a transaction, otherwise it MUST be clear.

D - LM (1 bit): A bit that specifies whether the message is the last one sent within the context of a transaction. This bit MUST be set if the message is the last one in a transaction, otherwise it MUST be clear.

ID (20 bits): An array of 20 bits that specifies an identifier to correlate this packet to the transaction under which it was captured. The message transfer protocols MUST generate an identifier for the transaction and assign all packets captured under the transaction to the value. This identifier MUST be unique across all such identifiers generated by the sender queue manager.

E - X1 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

F - X2 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

G - X3 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

H - X4 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

I - X5 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

J - X6 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

K - X7 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

L - X8 (1 bit): An unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

TxSequenceID (8 bytes): A transactional sequence identifier, as specified in TxSequenceID. This value identifies the transactional sequence that the **TxSequenceNumber** and **PreviousTxSequenceNumber** are within. For more details, see section 2.2.18.1.2.

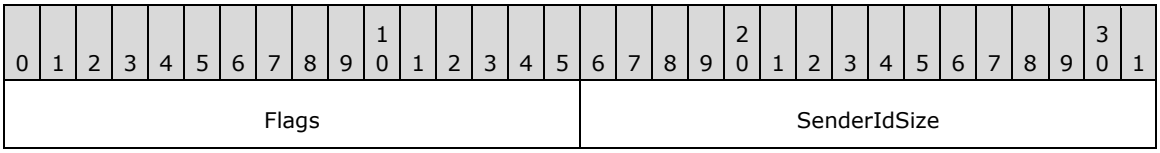
TxSequenceNumber (4 bytes): A 32-bit unsigned integer that is the message sequence number within the **TxSequenceID** sequence. This field MUST be set to the value that represents the message position within the transactional sequence. The first message within a sequence MUST be set to the value 1. This field has a valid range from 0x00000001 to 0xFFFFFFFF.

PreviousTxSequenceNumber (4 bytes): A 32-bit unsigned integer that is the sequence number of the previous message in the **TxSequenceID** sequence. This field MUST be set to the sequence number of the message that precedes this message in the transactional sequence. This value MUST be set to 0x00000000 if there is no previous message. This field has a valid range from 0x00000000 to 0xFFFFFFFE.

ConnectorQMGuid (16 bytes): An optional field containing an application-defined **GUID**, as specified in [MS-DTYP] section 2.3.4. If **Flags.CG** is set, this field MUST be present; otherwise, it MUST NOT. This field can be used by higher-layer messaging applications. The server MUST NOT process or interpret this field.

2.2.20.6 SecurityHeader

The optional SecurityHeader contains security information.



EncryptionKeySize	SignatureSize
SenderCertSize	
ProviderInfoSize	
SecurityData (variable)	
...	

Flags (2 bytes): A 16-bit unsigned short integer that contains a set of options that provides additional information about the packet. Any combination of these values is acceptable unless otherwise noted in the following table.

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
ST				A	B	C	D	AS				E	F	G	H																

ST (4 bits): Specifies the type of sender ID in the **SecurityData** field. This field MUST be set to one of the following values.

Value	Meaning
0x0	The **SecurityData.SecurityID** field is not present and the **SenderIdSize** field MUST be set to 0x0000.
0x1	The **SecurityData.SecurityID** field MUST contain the sender application security identifier (**SID**). The **SID** layout is specified in [MS-DTYP] section 2.4.2.2. The **SubAuthority** field of the **SID** packet is a variable-length array of unsigned 32-bit little-endian integers.
0x2	The **SecurityData.SecurityID** field MUST contain the queue manager **GUID**.

A - AU (1 bit): Indicates whether the message is authenticated. This field MUST be set to 0.

B - EB (1 bit): Indicates whether the body of the message is encrypted. If set, the **MessagePropertiesHeader.MessageBody** field MUST be encrypted by the sender and decrypted by the receiver.

For details about encryption on the sender side, see [MS-MQQB] section 3.1.7.1.5.

For details about decryption on the receiver side, see [MS-MQQB] section 3.1.5.8.3.

C - DE (1 bit): Indicates whether the default cryptographic provider is used.<19> When clear and **SignatureSize** is nonzero, the **SecurityData.ProviderName** MUST specify the name of the alternate provider.

D - AI (1 bit): Indicates whether the **SecurityData** field is present. If set, the header MUST include a **SecurityData** field.

AS (4 bits): Indicates the authentication signature type. This field MUST be set to 0.

E - X12 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

F - X13 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

G - X14 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

H - X15 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

SenderIdSize (2 bytes): A 16-bit unsigned integer that specifies the size of the **SecurityData.SecurityID** field. This value MUST be set to the size, in bytes, of the security identifier in the **SecurityData.SecurityID** field. This field has a valid range from 0x0000 to 0xFFFF, inclusive.

EncryptionKeySize (2 bytes): A 16-bit unsigned integer that specifies the size of the **SecurityData.EncryptionKey** field. This value MUST be set to the size, in bytes, of the encryption key in the **SecurityData.EncryptionKey** field. This field has a valid range from 0x0000 to 0xFFFF, inclusive.

SignatureSize (2 bytes): A 16-bit unsigned integer that specifies the size of the **SecurityData.Signature** field. This value MUST be set to the size, in bytes, of the sender signature in the **SecurityData.Signature** field. This field has a valid range from 0x0000 to 0xFFFF, inclusive.

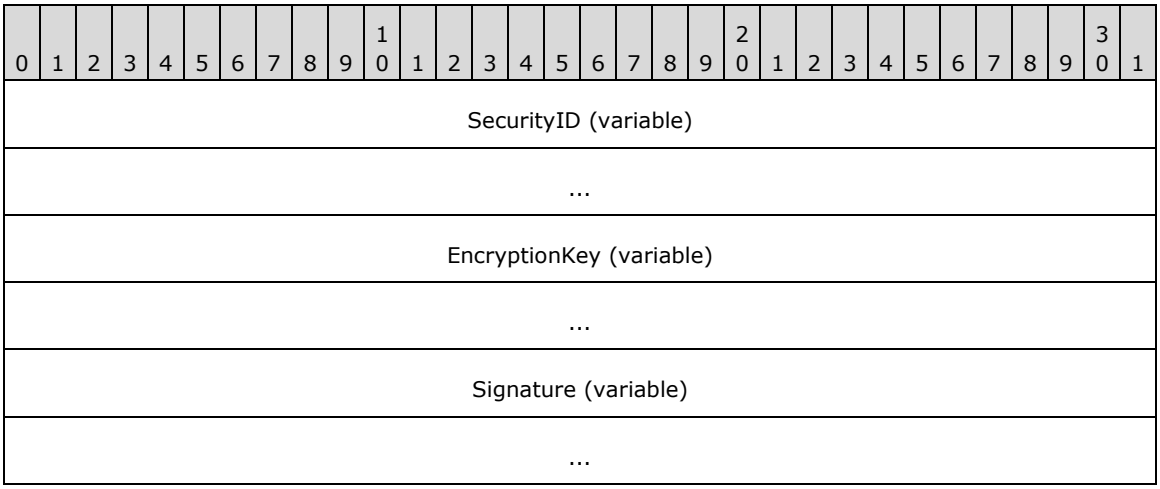
SenderCertSize (4 bytes): A 32-bit unsigned integer that specifies the size of the **SecurityData.SenderCert** field. This value MUST be set to the size, in bytes, of the sender signature in the **SecurityData.SenderCert** field. This field has a valid range from 0x00000000 to a value 0x0000FFFF, inclusive.

ProviderInfoSize (4 bytes): A 32-bit unsigned integer that specifies the size of the **SecurityData.ProviderInfo** field. This value MUST be set to the size, in bytes, of the security provider information in the **SecurityData.ProviderInfo** field. This field has a valid range between 0x00000000 and the size limit imposed by the value of **BaseHeader.PacketSize**.

At least one of the fields SenderIdSize, EncryptionKeySize, SignatureSize, SenderCertSize, and ProviderInfoSize MUST be nonzero.

SecurityData (variable): An optional variable-length array of bytes containing additional security information. This field MUST contain the security information specified in the **Flags** field.

The data appears in the order specified below. Each field MUST be aligned up to the next 4-byte boundary. The size of each field is specified by the corresponding **SenderIdSize**, **EncryptionKeySize**, **SignatureSize**, **SenderCertSize**, and **ProviderInfoSize** fields. An item with a size of zero occupies no space in the **SecurityData** array.



SenderCert (variable)
...
ProviderInfo (variable)
...

SecurityID (variable): Contains the sender SID or the sending queue manager GUID. This field MUST be set to the queue manager **GUID** when the packet is sent and signed by the queue manager.

EncryptionKey (variable): Sender symmetrical encryption key.

Signature (variable): The packet digital signature. The type of signature is specified by the MSMQ version as described in the following table and the hash algorithm is specified by the **MessagePropertiesHeader.HashAlgorithm** field.

MSMQ Version	Signature Type
MSMQ 1.0	The **SecurityData.Signature** field is an MSMQ 1.0 digital signature. If the **SecurityData.Flags.ST** field is set to 1, the **SecurityData.SecurityID** field MUST contain the sender application security identifier. If the **SecurityData.Flags.ST** field is set to 2, it specifies that the message is signed with Sender ID as the **Signature**. If set, the **SecurityData.SecurityID** field MUST contain the queue manager GUID. The signature MUST be a hash of the MSMQ 1.0 Digital Signature Properties (section 2.5.1).
MSMQ 2.0	The **SecurityData.Signature** field is an MSMQ 2.0 digital signature. The signature MUST be a hash of the MSMQ 2.0 Digital Signature Properties (section 2.5.2).
MSMQ 3.0, MSMQ 4.0, MSMQ 5.0, or MSMQ 6.0	The **SecurityData.Signature** field is an MSMQ 3.0 digital signature. The signature MUST be a hash of the MSMQ 3.0 Digital Signature Properties (section 2.5.3).

The hash algorithm that is used to compute the **SecurityData.Signature** field is specified by the **MessagePropertiesHeader.HashAlgorithm** field.

For details about signature and hash computations on the sender side, see [MS-MQQB] section 3.1.7.1.4.

For details about authentication on the receiver side, see [MS-MQQB] section 3.1.5.8.3.

SenderCert (variable): Sender X.509 digital certificate. Details are as specified in [RFC3280]. The public key that is contained in the certificate has the following structure.

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
0x06								0x02								0x00								Key Type							
...																								0x52							
0x53								0x41								0x31								Length in bits							

...	Public Exponent
...	Modulus (variable)
...	

Key Type (4 bytes): This MUST be set to 0x00002400 for RSA signing keys and 0x0000A400 for RSA encryption keys.

Length in bits (4 bytes): This 32-bit unsigned number MUST be the length of the RSA modulus. It MUST contain the length, in bits, of the Modulus field.

Public Exponent (4 bytes): This MUST be a 32-bit unsigned integer. It MUST be the public exponent of the RSA key pair, referred to as e in [RFC3447] section 2.

Modulus (variable): This MUST be the RSA modulus, referred to as defined in [RFC3447] section 2. This field MUST be a multiple of 8 bits in length and MUST append padding bits needed to ensure this requirement. Padding bits MUST be set to zero. The public key SHOULD<20> be stored in the directory.

ProviderInfo (variable): Contains the information of the alternative provider used to produce the signature.<21> If the Flags.DE bit is clear and the **ProviderInfoSize** is nonzero, this field MUST be set; otherwise it MUST NOT be included in the **SecurityData** field. The layout of this field is as follows.<22>

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
ProviderType																															
ProviderName (variable)																															
...																															

ProviderType (4 bytes): A 32-bit unsigned integer that indicates the type of the alternative provider used to produce the signature.

ProviderName (variable): A null-terminated Unicode string that contains the name of the alternative provider used to produce the signature.

2.2.20.7 SoapHeader

The optional SoapHeader packet contains application-defined information.

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
HeaderSectionID																Reserved															
HeaderDataLength																															
Header (variable)																															
...																															

BodySectionID	Reserved1
BodyDataLength	
Body (variable)	
...	

HeaderSectionID (2 bytes): A 16-bit unsigned short integer field that MUST be set to 0x0320.

Reserved (2 bytes): A 16-bit unsigned integer field that is reserved for future use. The sender SHOULD set this field to 0x0000, and the receiver MUST ignore it on receipt.

HeaderDataLength (4 bytes): A 32-bit unsigned integer that specifies the length of the **Header** field. This field MUST be set to the number of elements in the Unicode **Header** field, including the terminating null character. This field has a valid range between 0x00000000 and the size limit imposed by the value of BaseHeader.PacketSize.

Header (variable): A null-terminated Unicode string. This field contains an application-defined string.

BodySectionID (2 bytes): A 16-bit unsigned short integer that MUST be set to 0x0384.

Reserved1 (2 bytes): A 16-bit unsigned short integer field reserved for future use. The sender SHOULD set this field to 0x0000, and the receiver MUST ignore it on receipt.

BodyDataLength (4 bytes): A 32-bit unsigned integer that specifies the length of the **Body** field. This field MUST be set to the number of elements in the Unicode **Body** field, including the terminating null character. This field has a valid range between 0x00000000 and the size limit imposed by the value of BaseHeader.PacketSize.

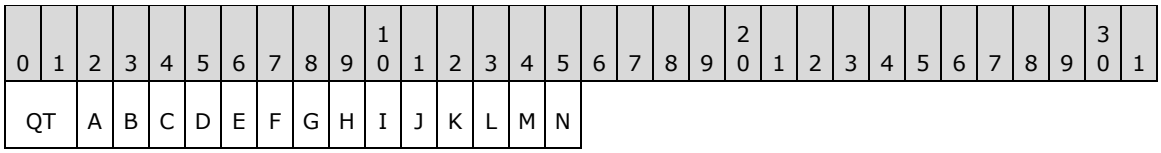
Body (variable): A null-terminated Unicode string. This field contains an application-defined string.

2.2.20.8 DebugHeader

The DebugHeader specifies the queue to receive trace messages for this UserMessage Packet. For details about how this header is used when tracing is enabled<23> see [MS-MQQB] section 3.1.5.8.9.

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
Flags																Reserved															
QueueIdentifier (16 bytes, optional)																															
...																															
...																															

Flags (2 bytes): A 16-bit unsigned short integer field that provides bit flags containing additional information about the packet.



QT (2 bits): Specifies the queue type. This field MUST be set to one of the following values.

Value	Meaning
0x0	No queue. The **QueueIdentifier** field is not present.
0x1	Public queue. The **QueueIdentifier** field contains a 16-byte queue **GUID**, as specified in [MS-DTYP] section 2.3.4.

A - X2 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

B - X3 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

C - X4 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

D - X5 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

E - X6 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

F - X7 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

G - X8 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

H - X9 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

I - X10 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

J - X11 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

K - X12 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

L - X13 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

M - X14 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

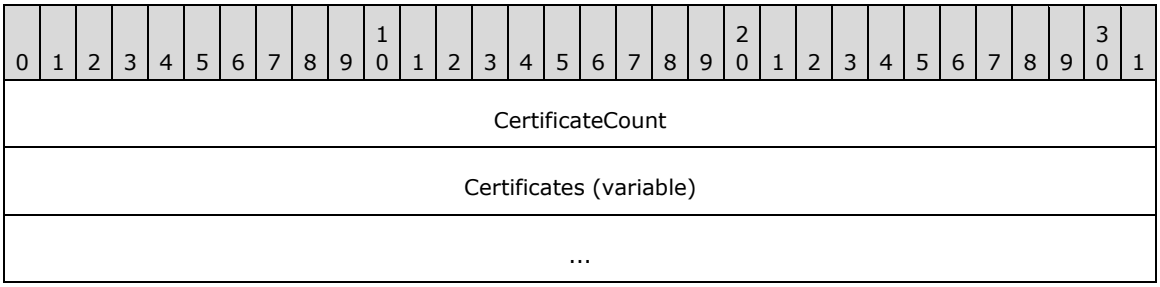
N - X15 (1 bit): Unused bit field. This field SHOULD NOT be set when sent and MUST be ignored on receipt.

Reserved (2 bytes): A 16-bit unsigned integer field that is reserved for future use. The sender SHOULD set this field to 0x0000, and the receiver MUST ignore it on receipt.

QueueIdentifier (16 bytes): An optional field that contains a **GUID**, as specified in [MS-DTYP] section 2.3.4, which is the identifier of the queue that is used to store trace messages. This field MUST be present when **DebugHeader.Flags.QT** is set to 0x1; otherwise, it MUST NOT be present.

2.2.21 MQUSERSIGNCERTS

The MQUSERSIGNCERTS structure defines a format for packing multiple MQUSERSIGNCERT structures into a single BLOB (section 2.2.15).

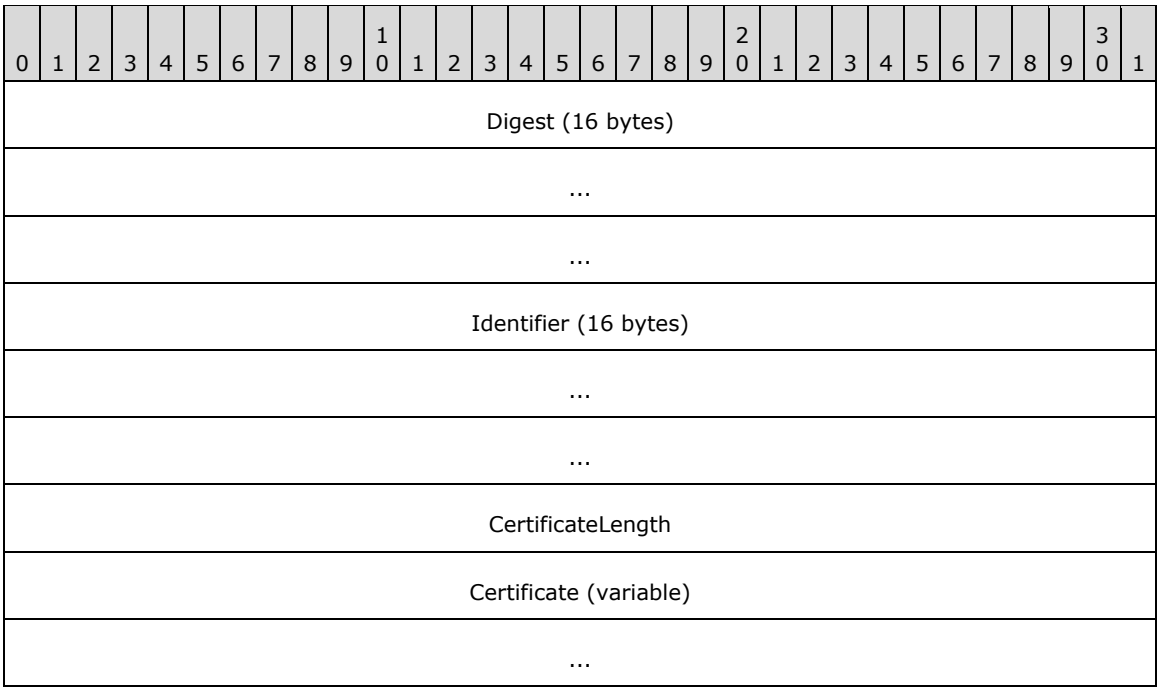


CertificateCount (4 bytes): A 32-bit unsigned integer that contains the count of MQUSERSIGNCERT structures in the **Certificates** field.

Certificates (variable): A variable-length array of bytes that contains MQUSERSIGNCERT structures.

2.2.22 MQUSERSIGNCERT

The MQUSERSIGNCERT structure defines one public key certificate stored in the user's signing certificate list (MQUSERSIGNCERTS (section 2.2.21)).



Digest (16 bytes): A 16-byte value that is computed as the MD5 hash of the user's X.509 certificate contained in the **Certificate** field, as defined in [RFC1321].

Identifier (16 bytes): A **GUID**, as specified in [MS-DTYP] section 2.3.4, that identifies the certificate contained in the **Certificate** field. It is generated by the entity that registers the certificate.

CertificateLength (4 bytes): A 32-bit unsigned integer that contains the size, in bytes, of the **Certificate** field.

Certificate (variable): A variable length array of bytes that contains an X.509-encoded certificate as defined in [RFC3280].

2.2.23 MQQMACCESSMASK

The MQQMACCESSMASK bitfield enumeration values can be used to set the value of an **ACCESS_MASK** structure ([MS-DTYP] section 2.4.3), which is used to constrain the permissions for a Queue Manager.

```
typedef  enum
{
  MQSEC_DELETE_DEADLETTER_MESSAGE = 0x00000001,
  MQSEC_PEEK_DEADLETTER_MESSAGE = 0x00000002,
  MQSEC_CREATE_QUEUE = 0x00000004,
  MQSEC_SET_MACHINE_PROPERTIES = 0x00000010,
  MQSEC_GET_MACHINE_PROPERTIES = 0x00000020,
  MQSEC_DELETE_JOURNAL_QUEUE_MESSAGE = 0x00000040,
  MQSEC_PEEK_JOURNAL_QUEUE_MESSAGE = 0x00000080,
  MQSEC_DELETE_MACHINE = 0x00010000,
  MQSEC_GET_MACHINE_PERMISSIONS = 0x00020000,
  MQSEC_CHANGE_MACHINE_PERMISSIONS = 0x00040000,
  MQSEC_TAKE_MACHINE_OWNERSHIP = 0x00080000,
  MQSEC_RECEIVE_DEADLETTER_MESSAGE = (MQSEC_DELETE_DEADLETTER_MESSAGE    +
MQSEC_PEEK_DEADLETTER_MESSAGE),
  MQSEC_RECEIVE_JOURNAL_QUEUE_MESSAGE = (MQSEC_DELETE_JOURNAL_QUEUE_MESSAGE    +
MQSEC_PEEK_JOURNAL_QUEUE_MESSAGE),
  MQSEC_MACHINE_GENERIC_READ = (MQSEC_GET_MACHINE_PROPERTIES    +
MQSEC_GET_MACHINE_PERMISSIONS    + MQSEC_RECEIVE_DEADLETTER_MESSAGE    +
MQSEC_RECEIVE_JOURNAL_QUEUE_MESSAGE),
  MQSEC_MACHINE_GENERIC_WRITE = (MQSEC_GET_MACHINE_PROPERTIES    +
MQSEC_GET_MACHINE_PERMISSIONS    + MQSEC_CREATE_QUEUE),
  MQSEC_MACHINE_GENERIC_ALL = (MQSEC_RECEIVE_DEADLETTER_MESSAGE    +
MQSEC_RECEIVE_JOURNAL_QUEUE_MESSAGE    + MQSEC_CREATE_QUEUE    + MQSEC_SET_MACHINE_PROPERTIES
+ MQSEC_GET_MACHINE_PROPERTIES    + MQSEC_DELETE_MACHINE    + MQSEC_GET_MACHINE_PERMISSIONS
+ MQSEC_CHANGE_MACHINE_PERMISSIONS    + MQSEC_TAKE_MACHINE_OWNERSHIP),
  MQSEC_MACHINE_WORLD_RIGHTS = (MQSEC_GET_MACHINE_PROPERTIES    +
MQSEC_GET_MACHINE_PERMISSIONS)
} MQQMACCESSMASK;
```

MQSEC_DELETE_DEADLETTER_MESSAGE: Specifies the permission required by a security principal to delete messages from the system dead-letter queue.

MQSEC_PEEK_DEADLETTER_MESSAGE: Specifies the permission required by a security principal to peek messages from the system dead-letter queue.

MQSEC_CREATE_QUEUE: Specifies the permission required by a security principal to create a new queue.

MQSEC_SET_MACHINE_PROPERTIES: Specifies the permission required by a security principal to change the properties of the queue manager.

MQSEC_GET_MACHINE_PROPERTIES: Specifies the permission required by a security principal to read the properties of the queue manager.

MQSEC_DELETE_JOURNAL_QUEUE_MESSAGE: Specifies the permission required by a security principal to delete a message from the system queue journal.

MQSEC_PEEK_JOURNAL_QUEUE_MESSAGE: Specifies the permission required by a security principal to peek a message from the system queue journal.

MQSEC_DELETE_MACHINE: Specifies the permission required by a security principal to delete the queue manager.

MQSEC_GET_MACHINE_PERMISSIONS: Specifies the permission required by a security principal to get the security descriptor of the queue manager.

MQSEC_CHANGE_MACHINE_PERMISSIONS: Specifies the permission required by a security principal to set or modify the security descriptor of the queue manager.

MQSEC_TAKE_MACHINE_OWNERSHIP: Specifies the permission required by a security principal to change the owner of the queue manager.

MQSEC_RECEIVE_DEADLETTER_MESSAGE: Specifies the permission required by a security principal to destructively read a message from the system dead-letter queue.

MQSEC_RECEIVE_JOURNAL_QUEUE_MESSAGE: Specifies the permission required by a security principal to destructively read a message from the system queue journal.

MQSEC_MACHINE_GENERIC_READ: Specifies the permission required by a security principal to read the queue manager properties, read the queue manager permissions, and destructively read a message from the system dead-letter or system queue journal.

MQSEC_MACHINE_GENERIC_WRITE: Specifies the permission required by a security principal to read the queue manager properties, read the queue manager permissions, and create new queues.

MQSEC_MACHINE_GENERIC_ALL: Specifies the permission required by a security principal to perform all of the operations listed previously.

MQSEC_MACHINE_WORLD_RIGHTS: Specifies the permission required by a security principal to get queue manager properties or permissions.

2.2.24 MQQUEUEACCESSMASK

The MQQUEUEACCESSMASK bitfield enumeration values can be used to set the value of an **ACCESS_MASK** structure ([MS-DTYP] section 2.4.3), which is used to constrain the permissions for a Queue.

```
typedef  enum
{
  MQSEC_DELETE_MESSAGE = 0x00000001,
  MQSEC_PEEK_MESSAGE = 0x00000002,
  MQSEC_WRITE_MESSAGE = 0x00000004,
  MQSEC_DELETE_JOURNAL_MESSAGE = 0x00000008,
  MQSEC_SET_QUEUE_PROPERTIES = 0x00000010,
  MQSEC_GET_QUEUE_PROPERTIES = 0x00000020,
  MQSEC_DELETE_QUEUE = 0x00010000,
  MQSEC_GET_QUEUE_PERMISSIONS = 0x00020000,
  MQSEC_CHANGE_QUEUE_PERMISSIONS = 0x00040000,
  MQSEC_TAKE_QUEUE_OWNERSHIP = 0x00080000,
  MQSEC_RECEIVE_MESSAGE = (MQSEC_DELETE_MESSAGE    | MQSEC_PEEK_MESSAGE),
  MQSEC_RECEIVE_JOURNAL_MESSAGE = (MQSEC_DELETE_JOURNAL_MESSAGE    | MQSEC_PEEK_MESSAGE),
  MQSEC_QUEUE_GENERIC_READ = (MQSEC_GET_QUEUE_PROPERTIES    | MQSEC_GET_QUEUE_PERMISSIONS
| MQSEC_RECEIVE_MESSAGE    | MQSEC_RECEIVE_JOURNAL_MESSAGE),
  MQSEC_QUEUE_GENERIC_WRITE = (MQSEC_GET_QUEUE_PROPERTIES    | MQSEC_GET_QUEUE_PERMISSIONS
| MQSEC_WRITE_MESSAGE),
  MQSEC_QUEUE_GENERIC_ALL = (MQSEC_RECEIVE_MESSAGE    | MQSEC_RECEIVE_JOURNAL_MESSAGE    |
MQSEC_WRITE_MESSAGE | MQSEC_SET_QUEUE_PROPERTIES    | MQSEC_GET_QUEUE_PROPERTIES    |
MQSEC_DELETE_QUEUE    | MQSEC_GET_QUEUE_PERMISSIONS    | MQSEC_CHANGE_QUEUE_PERMISSIONS    |
MQSEC_TAKE_QUEUE_OWNERSHIP)
```

```
} MQQUEUEACCESSMASK;
```

MQSEC_DELETE_MESSAGE: Specifies the permission required by a security principal to delete messages from the queue.

MQSEC_PEEK_MESSAGE: Specifies the permission required by a security principal to peek messages from the queue.

MQSEC_WRITE_MESSAGE: Specifies the permission required by a security principal to insert messages into the queue.

MQSEC_DELETE_JOURNAL_MESSAGE: Specifies the permission required by a security principal to delete messages from the queue journal associated with the queue.

MQSEC_SET_QUEUE_PROPERTIES: Specifies the permission required by a security principal to modify the properties of the queue.

MQSEC_GET_QUEUE_PROPERTIES: Specifies the permission required by a security principal to read the properties of the queue.

MQSEC_DELETE_QUEUE: Specifies the permission required by a security principal to delete a queue.

MQSEC_GET_QUEUE_PERMISSIONS: Specifies the permission required by a security principal to read the permissions for the queue.

MQSEC_CHANGE_QUEUE_PERMISSIONS: Specifies the permission required by a security principal to modify the permissions for the queue.

MQSEC_TAKE_QUEUE_OWNERSHIP: Specifies the permission required by a security principal to change the owner for the queue.

MQSEC_RECEIVE_MESSAGE: Specifies the permission required by a security principal to destructively read a message from the queue.

MQSEC_RECEIVE_JOURNAL_MESSAGE: Specifies the permission required by a security principal to destructively read a message from the queue journal associated with the queue.

MQSEC_QUEUE_GENERIC_READ: Specifies the permission required by a security principal to read the queue properties, read the queue permissions, and destructively receive messages from the queue or the associated queue journal.

MQSEC_QUEUE_GENERIC_WRITE: Specifies the permission required by a security principal to read the queue properties, read the queue permissions, and insert messages into the queue.

MQSEC_QUEUE_GENERIC_ALL: Specifies the permission required by a security principal to perform all of the operations listed previously.

2.2.25 MQSITEACCESSMASK

The MQSITEACCESSMASK bitfield enumeration values can be used to set the value of an **ACCESS_MASK** structure ([MS-DTYP] section 2.4.3), which is used to constrain the permissions for a Site.

```
typedef  enum
{
  MQSEC_CREATE_FRS = 0x00000001,
  MQSEC_CREATE_BSC = 0x00000002,
  MQSEC_CREATE_MACHINE = 0x00000004,
  MQSEC_SET_SITE_PROPERTIES = 0x00000010,
```

```
  MQSEC_GET_SITE_PROPERTIES = 0x00000020,
  MQSEC_DELETE_SITE = 0x00010000,
  MQSEC_GET_SITE_PERMISSIONS = 0x00020000,
  MQSEC_CHANGE_SITE_PERMISSIONS = 0x00040000,
  MQSEC_TAKE_SITE_OWNERSHIP = 0x00080000,
  MQSEC_SITE_GENERIC_READ = (MQSEC_GET_SITE_PROPERTIES    + MQSEC_GET_SITE_PERMISSIONS),
  MQSEC_SITE_GENERIC_WRITE = (MQSEC_GET_SITE_PROPERTIES    + MQSEC_GET_SITE_PERMISSIONS    +
MQSEC_CREATE_MACHINE),
  MQSEC_SITE_GENERIC_ALL = (MQSEC_CREATE_FRS    + MQSEC_CREATE_BSC    + MQSEC_CREATE_MACHINE
+ MQSEC_SET_SITE_PROPERTIES    + MQSEC_GET_SITE_PROPERTIES    + MQSEC_DELETE_SITE    +
MQSEC_GET_SITE_PERMISSIONS    + MQSEC_CHANGE_SITE_PERMISSIONS    + MQSEC_TAKE_SITE_OWNERSHIP)
} MQSITEACCESSMASK;
```

MQSEC_CREATE_FRS: Specifies the permission required by a security principal to add a routing server to the site.

MQSEC_CREATE_BSC: Specifies the permission required by a security principal to create a Backup Site Controller (BSC) for the site.

MQSEC_CREATE_MACHINE: Specifies the permission required by a security principal to add a queue manager to the site.

MQSEC_SET_SITE_PROPERTIES: Specifies the permission required by a security principal to modify properties of the site.

MQSEC_GET_SITE_PROPERTIES: Specifies the permission required by a security principal to read properties of the site.

MQSEC_DELETE_SITE: Specifies the permission required by a security principal to delete the site.

MQSEC_GET_SITE_PERMISSIONS: Specifies the permission required by a security principal to read permissions for the site.

MQSEC_CHANGE_SITE_PERMISSIONS: Specifies the permission required by a security principal to modify permissions for the site.

MQSEC_TAKE_SITE_OWNERSHIP: Specifies the permission required by a security principal to modify the owner of the site.

MQSEC_SITE_GENERIC_READ: Specifies the permission required by a security principal to read the properties and permissions for the site.

MQSEC_SITE_GENERIC_WRITE: Specifies the permission required by a security principal to read properties, read permissions, and add queue managers to the site.

MQSEC_SITE_GENERIC_ALL: Specifies the permission required by a security principal to perform all of the operations listed previously.

2.2.26 MQENTACCESSMASK

The MQENTACCESSMASK bitfield enumeration values can be used to set the value of an **ACCESS_MASK** structure ([MS-DTYP] section 2.4.3), which is used to constrain the permissions for an Enterprise.

```
typedef  enum
{
  MQSEC_CREATE_USER = 0x00000001,
  MQSEC_CREATE_SITE = 0x00000002,
  MQSEC_CREATE_CN = 0x00000004,
  MQSEC_SET_ENTERPRISE_PROPERTIES = 0x00000010,
  MQSEC_GET_ENTERPRISE_PROPERTIES = 0x00000020,
```

```
  MQSEC_DELETE_ENTERPRISE = 0x00010000,
  MQSEC_GET_ENTERPRISE_PERMISSIONS = 0x00020000,
  MQSEC_CHANGE_ENTERPRISE_PERMISSIONS = 0x00040000,
  MQSEC_TAKE_ENTERPRISE_OWNERSHIP = 0x00080000,
  MQSEC_ENTERPRISE_GENERIC_READ = (MQSEC_CREATE_USER    + MQSEC_GET_ENTERPRISE_PROPERTIES
+ MQSEC_GET_ENTERPRISE_PERMISSIONS),
  MQSEC_ENTERPRISE_GENERIC_WRITE = (MQSEC_CREATE_USER    + MQSEC_GET_ENTERPRISE_PROPERTIES
+ MQSEC_GET_ENTERPRISE_PERMISSIONS    + MQSEC_CREATE_SITE    + MQSEC_CREATE_CN),
  MQSEC_ENTERPRISE_GENERIC_ALL = (MQSEC_CREATE_USER    + MQSEC_CREATE_CN    +
MQSEC_CREATE_SITE    + MQSEC_SET_ENTERPRISE_PROPERTIES    + MQSEC_GET_ENTERPRISE_PROPERTIES
+ MQSEC_DELETE_ENTERPRISE    + MQSEC_GET_ENTERPRISE_PERMISSIONS    +
MQSEC_CHANGE_ENTERPRISE_PERMISSIONS    + MQSEC_TAKE_ENTERPRISE_OWNERSHIP)
} MQENTACCESSMASK;
```

MQSEC_CREATE_USER: Specifies the permission required by a security principal to add a user to the enterprise.

MQSEC_CREATE_SITE: Specifies the permission required by a security principal to create a site in the enterprise.

MQSEC_CREATE_CN: Specifies the permission required by a security principal to create a connected network in the enterprise.

MQSEC_SET_ENTERPRISE_PROPERTIES: Specifies the permission required by a security principal to modify properties of the enterprise.

MQSEC_GET_ENTERPRISE_PROPERTIES: Specifies the permission required by a security principal to read properties of the enterprise.

MQSEC_DELETE_ENTERPRISE: Specifies the permission required by a security principal to delete the enterprise.

MQSEC_GET_ENTERPRISE_PERMISSIONS: Specifies the permission required by a security principal to read permissions for the enterprise.

MQSEC_CHANGE_ENTERPRISE_PERMISSIONS: Specifies the permission required by a security principal to modify permissions for the enterprise.

MQSEC_TAKE_ENTERPRISE_OWNERSHIP: Specifies the permission required by a security principal to modify the owner of the enterprise.

MQSEC_ENTERPRISE_GENERIC_READ: Specifies the permission required by a security principal to read the properties and permissions for the enterprise, and create users.

MQSEC_ENTERPRISE_GENERIC_WRITE: Specifies the permission required by a security principal to read properties and permissions, and to add users, connected networks, and sites to the enterprise.

MQSEC_ENTERPRISE_GENERIC_ALL: Specifies the permission required by a security principal to perform all of the operations listed previously.

2.2.27 MQCNACCESSMASK

The MQCNACCESSMASK bitfield enumeration values can be used to set the value of an **ACCESS_MASK** structure ([MS-DTYP] section 2.4.3), which is used to constrain the permissions for a Connected Network.

```
typedef  enum
{
  MQSEC_CN_OPEN_CONNECTOR = 0x00000001,
  MQSEC_SET_CN_PROPERTIES = 0x00000010,
  MQSEC_GET_CN_PROPERTIES = 0x00000020,
```

```
  MQSEC_DELETE_CN = 0x00010000,
  MQSEC_GET_CN_PERMISSIONS = 0x00020000,
  MQSEC_CHANGE_CN_PERMISSIONS = 0x00040000,
  MQSEC_TAKE_CN_OWNERSHIP = 0x00080000,
  MQSEC_CN_GENERIC_READ = (MQSEC_GET_CN_PROPERTIES    + MQSEC_GET_CN_PERMISSIONS),
  SEC_CN_GENERIC_ALL = (MQSEC_CN_OPEN_CONNECTOR    + MQSEC_SET_CN_PROPERTIES    +
MQSEC_GET_CN_PROPERTIES    + MQSEC_DELETE_CN    + MQSEC_GET_CN_PERMISSIONS    +
MQSEC_CHANGE_CN_PERMISSIONS    + MQSEC_TAKE_CN_OWNERSHIP)
} MQCNACCESSMASK;
```

MQSEC_CN_OPEN_CONNECTOR: Specifies the permission required by a security principal to open a connector queue in the connected network.

MQSEC_SET_CN_PROPERTIES: Specifies the permission required by a security principal to modify properties of the connected network.

MQSEC_GET_CN_PROPERTIES: Specifies the permission required by a security principal to read properties of the connected network.

MQSEC_DELETE_CN: Specifies the permission required by a security principal to delete the connected network.

MQSEC_GET_CN_PERMISSIONS: Specifies the permission required by a security principal to read permissions for the connected network.

MQSEC_CHANGE_CN_PERMISSIONS: Specifies the permission required by a security principal to modify permissions for the site.

MQSEC_TAKE_CN_OWNERSHIP: Specifies the permission required by a security principal to modify the owner of the connected network.

MQSEC_CN_GENERIC_READ: Specifies the permission required by a security principal to read properties and permissions of the connected network.

SEC_CN_GENERIC_ALL: Specifies the permission required by a security principal to perform all of the operations listed previously.

2.3 PROPID

When making MSMQ-related API function calls, object properties are specified by providing an array of property identifiers (a unique PROPID value). The associated property values are specified (or returned) in a related array of PROPVARIANT structures. The values (in decimal), their PROPVARIANT types (as specified in section 2.2.12), and their associated symbolic names are listed in the PROPID subsections. Related properties are grouped together within each PROPID subsection.

A PROPID is an unsigned 32-bit value.

This type is declared as follows:

```
typedef unsigned long PROPID;
```

Each directory object type and management type has a set of properties associated with it. The following sections define the property identifier ranges and the properties associated within each range.

Unless otherwise specified, properties are valid for all MSMQ versions.

2.3.1 Queue Property Identifiers

Queue properties specify attributes of individual queue objects.

2.3.1.1 PROPID_Q_INSTANCE

Value: 101

Variant type: VT_CLSID

Description: GUID for the queue.

2.3.1.2 PROPID_Q_TYPE

Value: 102

Variant type: VT_CLSID

Description: A user-defined value that indicates the type of service that the queue provides. The value is optionally specified at queue creation and can be changed after the queue has been created.

2.3.1.3 PROPID_Q_PATHNAME

Value: 103

Variant type: VT_LPWSTR

Description: The path of the queue. The value is specified at queue creation and is immutable thereafter. The value MUST conform to the ABNF rule QueuePathName, as specified in section 2.1.1.

2.3.1.4 PROPID_Q_JOURNAL

Value: 104

Variant type: VT_UI1

Description: A value that specifies how MSMQ tracks messages removed from the queue. This field MUST be one of the following.

Value	Constant	Description
0	MQ_JOURNAL_NONE	The default. Target journaling is not requested. Messages removed from the destination queue are no longer available.
1	MQ_JOURNAL	Target journaling is requested. Copies of messages are stored in the journal of the queue whenever a receiving application removes a message.

2.3.1.5 PROPID_Q_QUOTA

Value: 105

Variant type: VT_UI4

Description: Maximum size (in kilobytes) of a queue.<24>

2.3.1.6 PROPID_Q_BASEPRIORITY

Value: 106

Variant type: VT_I2

Description: Priority level of the queue. PROPID_Q_BASEPRIORITY applies only to public queues that can be located through the directory service (using a public format name). The base priority of private queues, as well as public queues accessed directly, is always 0x0000. Any attempt to create this property and set its value or to set the value of an existing property for a private queue when the queue is being created or after the queue is created will be ignored and will cause no change. The value MUST be set to a valid priority level. Priority levels are integer values between -32768 (0x8000) and +32767 (0x7fff). The default priority level is 0x0000.

2.3.1.7 PROPID_Q_JOURNAL_QUOTA

Value: 107

Variant type: VT_UI4

Description: Maximum size (in kilobytes) of the queue journal. Valid values are in the range 0 to 0xffffffff.<25>

2.3.1.8 PROPID_Q_LABEL

Value: 108

Variant type: VT_LPWSTR

Description: A descriptive label (maximum 124 characters) for the queue.

2.3.1.9 PROPID_Q_CREATE_TIME

Value: 109

Variant type: VT_I4

Description: The time when the queue was created. Time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

2.3.1.10 PROPID_Q_MODIFY_TIME

Value: 110

Variant type: VT_I4

Description: The time when the queue properties were last modified. The time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

2.3.1.11 PROPID_Q_AUTHENTICATE

Value: 111

Variant type: VT_UI1

Description: Authentication level of the queue. MUST be one of the following values.

Value	Constant
0x0	MQ_AUTHENTICATE_NONE
0x01	MQ_AUTHENTICATE

2.3.1.12 PROPID_Q_PRIV_LEVEL

Value: 112

Variant type: VT_UI4

Description: Privacy level of the queue. MUST be one of the following values.

Value	Constant
0x00000000	MQ_PRIV_LEVEL_NONE
0x00000001	MQ_PRIV_LEVEL_OPTIONAL
0x00000002	MQ_PRIV_LEVEL_BODY

MQ_PRIV_LEVEL_NONE: The queue accepts only nonprivate (clear) messages.

MQ_PRIV_LEVEL_BODY: The queue accepts only private (encrypted) messages.

MQ_PRIV_LEVEL_OPTIONAL: The default. The queue does not enforce privacy. It accepts private (encrypted) messages and nonprivate (clear) messages.

2.3.1.13 PROPID_Q_TRANSACTION

Value: 113

Variant type: VT_UI1

Description: Transaction level of the queue. MUST be one of the following values.

Value	Constant
0x00	MQ_TRANSACTIONAL_NONE
0x01	MQ_TRANSACTIONAL

2.3.1.14 PROPID_Q_SCOPE

Value: 114

Variant type: VT_UI1

Description: A value that specifies the scope of a queue object. The value MUST be one of the following.

Value	Constant	Description
0x00	MQDS_SITESCOPE	Indicates a site scope.
0x01	MQDS_ENTERPRISESCOPE	Indicates an enterprise scope.

2.3.1.15 PROPID_Q_QMID

Value: 115

Variant type: VT_CLSID

Description: Contains the GUID of the queue manager that hosts the queue.

2.3.1.16 PROPID_Q_PARTITIONID

Value: 116

Variant type: VT_CLSID

Description: This property MAY<26> be used to group MSMQ directory objects.

2.3.1.17 PROPID_Q_SEQNUM

Value: 117

Variant type: VT_BLOB

Description: Contains the sequence number of the queue object.

2.3.1.18 PROPID_Q_HASHKEY

Value: 118

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.1.19 PROPID_Q_LABEL_HASHKEY

Value: 119

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.1.20 PROPID_Q_FULL_PATH

Value: 121

Variant type: VT_LPWSTR

Description: Contains the distinguished name (DN) of the queue object in Active Directory (as specified in [MS-ADTS]).

Note This property identifier was introduced in MSMQ 2.0.

2.3.1.21 PROPID_Q_NAME_SUFFIX

Value: 123

Variant type: VT_LPWSTR

Description: Contains the suffix of the queue name if the name exceeds 64 characters (the length of the **Common-Name** attribute in Active Directory).

Note This property identifier was introduced in MSMQ 2.0.

2.3.1.22 PROPID_Q_PATHNAME_DNS

Value: 124

Variant type: VT_LPWSTR

Description: Contains the fully qualified domain name (FQDN) prefixed path of the queue. The value MUST conform to the ABNF for QueuePathName (as specified in section 2.1.1), where the computer name is the FQDN of the hosting computer.

Note This property identifier was introduced in MSMQ 2.0.

2.3.1.23 PROPID_Q_MULTICAST_ADDRESS

Value: 125

Variant type: VT_LPWSTR

Description: IP multicast address associated with the queue. The property value MUST contain a string that contains a valid multicast address conforming to the ABNF.

```
MulticastAddress = Address ":" Port
```

The ABNF rules for Address and Port are defined in section 2.1.6. The address MUST be in the class D range from 224.0.0.0 to 239.255.255.255. However, only certain ranges of addresses in this range are unreserved and available for sending multicast messages. For more information and the current list of reserved multicast addresses, see [IANAIMA]. There are no restrictions on the port number.

Note This property identifier was introduced in MSMQ 3.0.

2.3.1.24 PROPID_Q_ADS_PATH

Value: 126

Variant type: VT_LPWSTR

Description: Contains the Active Directory path to the public queue object stored in Active Directory. The value MUST conform to the ABNF for ldapurl (as specified in [RFC4516]).

The following example shows a possible Active Directory path of the queue "MyComp\MyQueue".

LDAP://MyLDAPServer/CN=MyQueue,CN=msmq,CN=MyComp,CN=Computers,DC=MyDomain,DC=MyCompany,DC=COM

Note This property identifier was introduced in MSMQ 3.0.

2.3.1.25 PROPID_Q_SECURITY

Value: 1101

Variant type: VT_BLOB

Description: Contains the security descriptor of the queue object. The BLOB layout is that of SECURITY_DESCRIPTOR, as specified in [MS-DTYP] section 2.4.6.<27>

2.3.1.26 PROPID_Q_OBJ_SECURITY

Value: 1102

Variant type: VT_BLOB

Description: Contains the security descriptor of the queue object. The BLOB layout is that of SECURITY_DESCRIPTOR (as specified in [MS-DTYP] section 2.4.6).

Note This property identifier was introduced in MSMQ 2.0.

2.3.1.27 PROPID_Q_SECURITY_INFORMATION

Value: 1103

Variant type: VT_UI4

Description: Contains options related to setting or retrieving a security descriptor. It contains SECURITY_INFORMATION (section 2.2.3).

2.3.2 Machine Property Identifiers

Machine object property identifiers describe a queue manager.

2.3.2.1 PROPID_QM_SITE_ID

Value: 201

Variant type: VT_CLSID

Description: Contains the site identifier GUID of the site in which the queue manager is located.

2.3.2.2 PROPID_QM_MACHINE_ID

Value: 202

Variant type: VT_CLSID

Description: A GUID that uniquely identifies the queue manager for the computer.

2.3.2.3 PROPID_QM_PATHNAME

Value: 203

Variant type: VT_LPWSTR

Description: The name of the computer where the queue manager is located.

2.3.2.4 PROPID_QM_ENCRYPTION_PK

Value: 205

Variant type: VT_UI1 | VT_VECTOR

Description: The public encryption key of the computer. This property is superseded by PROPID_QM_ENCRYPTION_PK_BASE if present.

2.3.2.5 PROPID_QM_ADDRESS

Value: 206

Variant type: VT_BLOB

Description: The network address or addresses of the computer. The blob layout is a packed array of TA_ADDRESS (section 2.2.4) structures.

2.3.2.6 PROPID_QM_CNS

Value: 207

Variant type: VT_CLSID | VT_VECTOR

Description: Contains an array of Connected Network identifiers for the connected networks that the queue manager supports.

2.3.2.7 PROPID_QM_OUTFRS

Value: 208

Variant type: VT_CLSID | VT_VECTOR

Description: An array of GUIDs of routing servers that act as outgoing interfaces for all MSMQ messages that a given machine sends.

2.3.2.8 PROPID_QM_INFRS

Value: 209

Variant type: VT_CLSID | VT_VECTOR

Description: An array of GUIDs of routing servers that act as incoming interfaces for all MSMQ messages that a given machine receives.

2.3.2.9 PROPID_QM_SERVICE

Value: 210

Variant type: VT_UI4

Description: Indicates the type of service that a given machine supports. The possible values are as follows.

Value	Meaning
0x00000000	The machine does not support any service.
0x00000001	The machine is an MSMQ Routing Server.
0x00000002	The machine is a Backup Site Controller (BSC).
0x00000004	The machine is a Primary Site Controller (PSC).
0x00000008	The machine is a Primary Enterprise Controller (PEC).
0x00000010	The machine is a RAS server.

2.3.2.10 PROPID_QM_QUOTA

Value: 214

Variant type: VT_UI4

Description: The disk quota for all queues located at the queue manager. Valid range: 0 to max unsigned 32-bit (0xffffffff).

2.3.2.11 PROPID_QM_PARTITIONID

Value: 211

Variant type: VT_CLSID

Description: This property MAY<28> be used to group MSMQ directory objects.

2.3.2.12 PROPID_QM_HASHKEY

Value: 212

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.2.13 PROPID_QM_SEQNUM

Value: 213

Variant type: VT_BLOB

Description: Contains the sequence number of the queue manager object.

2.3.2.14 PROPID_QM_JOURNAL_QUOTA

Value: 215

Variant type: VT_UI4

Description: Contains the systemwide journal storage quota, in kilobytes. Range restrictions are identical to PROPID_QM_QUOTA (section 2.3.2.10).

2.3.2.15 PROPID_QM_MACHINE_TYPE

Value: 216

Variant type: VT_LPWSTR

Description: A description of the operating system version and the MSMQ version. MAY be an empty string<29> or a version string<30>.

2.3.2.16 PROPID_QM_CREATE_TIME

Value: 217

Variant type: VT_I4

Description: The time when the directory object was created. Time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

2.3.2.17 PROPID_QM_MODIFY_TIME

Value: 218

Variant type: VT_I4

Description: The time when the directory object was last modified. The time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

2.3.2.18 PROPID_QM_FOREIGN

Value: 219

Variant type: VT_UI1

Description: Indicates whether the queue manager is a foreign system that services foreign queues. The value MUST be one of the following.

Constant	Value
FOREIGN_MACHINE	0x01
MSMQ_MACHINE	0x00

2.3.2.19 PROPID_QM_OS

Value: 220

Variant type: VT_UI4

Description: A value indicating the operating system type of the queue manager. The value MUST be one of the following.

String	Value	Description
MSMQ_OS_NONE	0x00000000	Unknown operating system type
MSMQ_OS_FOREIGN	0x00000100	Not a Windows operating system type

String	Value	Description
MSMQ_OS_95	0x00000200	Windows 95 operating system
MSMQ_OS_NTW	0x00000300	A client release of Windows
MSMQ_OS_NTS	0x00000400	A server release of Windows
MSMQ_OS_NTE	0x00000500	Windows Server 2003 Enterprise Edition operating system

2.3.2.20 PROPID_QM_FULL_PATH

Value: 221

Variant type: VT_LPWSTR

Description: The distinguishedName for the MSMQ Configuration object. The name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

Note Not valid for MSMQ 1.0.

2.3.2.21 PROPID_QM_SITE_IDS

Value: 222

Variant type: VT_CLSID | VT_VECTOR

Description: Contains an array of site identifiers for sites to which the computer belongs.

Note Not valid for MSMQ 1.0.

2.3.2.22 PROPID_QM_OUTFRS_DN

Value: 223

Variant type: VT_LPWSTR | VT_VECTOR

Description: An array of distinguished names for MSMQ routing servers through which all outgoing traffic for this computer will be routed. Each name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

Note Not valid for MSMQ 1.0.

2.3.2.23 PROPID_QM_INFRS_DN

Value: 224

Variant type: VT_LPWSTR | VT_VECTOR

Description: An array of distinguished names for MSMQ routing servers through which all incoming traffic to this computer will be routed. Each name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

Note Not valid for MSMQ 1.0.

2.3.2.24 PROPID_QM_SERVICE_ROUTING

Value: 227

Variant type: VT_UI1

Description: Indicates whether the queue manager is configured as a routing server. This value SHOULD be settable only by the MSMQ installer. The value MUST be one of the following.

Value	Meaning
0x00	The queue manager is NOT configured as a routing server.
0x01	The queue manager is configured as a routing server.

Note Not valid for MSMQ 1.0.

2.3.2.25 PROPID_QM_SERVICE_DSSERVER

Value: 228

Variant type: VT_UI1

Description: Indicates whether the installed version of Microsoft Message Queuing (MSMQ) provides MSMQ Directory Service (MQDS) services. This property value is stored in Active Directory as a Boolean.

2.3.2.26 PROPID_QM_SERVICE_DEPCLIENTS

Value: 229

Variant type: VT_UI1

Description: Indicates whether the installed version of Microsoft Message Queuing (MSMQ) acts as an MSMQ supporting server. This property value is stored in Active Directory as a Boolean.

2.3.2.27 PROPID_QM_ENCRYPTION_PK_BASE

Value: 231

Variant type: VT_UI1 | VT_VECTOR

Description: Contains the public encryption key of the computer.

Note Not valid for MSMQ 1.0.

2.3.2.28 PROPID_QM_ENCRYPTION_PK_ENHANCED

Value: 232

Variant type: VT_UI1 | VT_VECTOR

Description: Contains the enhanced (128-bit) public encryption key of the computer.

Note Not valid for MSMQ 1.0.

2.3.2.29 PROPID_QM_PATHNAME_DNS

Value: 233

Variant type: VT_LPWSTR

Description: Contains the FQDN of the computer.

Note Not valid for MSMQ 1.0.

2.3.2.30 PROPID_QM_OBJ_SECURITY

Value: 234

Variant type: VT_BLOB

Description: Contains the security descriptor of the MSMQ Configuration object. The BLOB layout is that of SECURITY_DESCRIPTOR, as specified in [MS-DTYP] section 2.4.6.

Note Not valid for MSMQ 1.0.

2.3.2.31 PROPID_QM_SECURITY_INFORMATION

Value: 237

Variant type: VT_UI4

Description: Contains options related to setting or retrieving a security descriptor. The value MUST conform to SECURITY_INFORMATION (section 2.2.3).

2.3.2.32 PROPID_QM_ENCRYPT_PKS

Value: 238

Variant type: VT_BLOB

Description: The computer's public key certificates used for signing formatted as an MQDSPUBLICKEYS (section 2.2.2) structure.

2.3.2.33 PROPID_QM_SIGN_PKS

Value: 239

Variant type: VT_BLOB

Description: The computer's public key certificates used for signing, formatted as an MQDSPUBLICKEYS (section 2.2.2) structure.

2.3.2.34 PROPID_QM_OWNER_SID

Value: 241

Variant type: VT_BLOB

Description: Contains the SID of the user who ran the setup program. It is passed from the MSMQ service that created the MSMQ Configuration object so that the server can add it with full control to the DACL of the newly created object. The **SID** layout is specified in [MS-DTYP] section 2.4.2.2. The

SubAuthority field of the **SID** packet is a variable-length array of unsigned 32-bit little-endian integers.

Note This property identifier was introduced in MSMQ 2.0.

2.3.2.35 PROPID_QM_GROUP_IN_CLUSTER

Value: 242

Variant type: VT_UI1

Description: Indicates that the MSMQ installation is in a group that is part of a cluster.

Used when creating the MSMQ Configuration objects. The value MUST be one of the following.

Constant	Value
MSMQ_GROUP_NOT_IN_CLUSTER	0x00
MSMQ_GROUP_IN_CLUSTER	0x01

Note This property identifier was introduced in MSMQ 2.0.

2.3.2.36 PROPID_QM_SECURITY

Value: 1201

Variant type: VT_BLOB

Description: Contains the security descriptor of the machine object. The layout of the **BLOB** is specified in [MS-DTYP] section 2.4.6.<31>

2.3.2.37 PROPID_QM_SIGN_PK

Value: 1202

Variant type: VT_BLOB

Description: The computer's public key certificates used for signing, formatted as an MQDSPUBLICKEYS (section 2.2.2) structure. This property can be specified only at object creation time.

2.3.2.38 PROPID_QM_ENCRYPT_PK

Value: 1203

Variant type: VT_BLOB

Description: The computer's public key certificates used for encryption, formatted as an MQDSPUBLICKEYS (section 2.2.2) structure. This property can be specified only at object creation time.

2.3.2.39 PROPID_QM_UPGRADE_DACL

Value: 1205

Variant type: VT_BLOB

Description: A dummy PROPID. It is used only in a set property operation to request that the PEC update the DACL of the calling computer. The **BLOB** MAY be empty. The server MUST ignore the value.

2.3.3 Site Property Identifiers

Site property identifiers pertain to the site object.

2.3.3.1 PROPID_S_PATHNAME

Value: 301

Variant type: VT_LPWSTR

Description: Contains the name of the site.

2.3.3.2 PROPID_S_SITEID

Value: 302

Variant type: VT_CLSID

Description: Contains the identifier of the site.

2.3.3.3 PROPID_S_GATES

Value: 303

Variant type: VT_CLSID | VT_VECTOR

Description: Contains the GUIDs of the MSMQ Configuration objects of the MSMQ queue managers that are the gates for this site.

2.3.3.4 PROPID_S_PSC

Value: 304

Variant type: VT_LPWSTR

Description: Contains the computer name of the PSC for the site.

2.3.3.5 PROPID_S_INTERVAL1

Value: 305

Variant type: VT_UI2

Description: In MSMQ mixed-mode, the default replication time (in seconds) within an MSMQ Site. The default is 2 seconds.

2.3.3.6 PROPID_S_INTERVAL2

Value: 306

Variant type: VT_UI2

Description: In MSMQ mixed-mode, the default replication time (in seconds) between MSMQ sites. The default is 10 seconds.

2.3.3.7 PROPID_S_PARTITIONID

Value: 307

Variant type: VT_CLSID

Description: This property MAY <32> be used to group MSMQ directory objects.

2.3.3.8 PROPID_S_SEQNUM

Value: 308

Variant type: VT_BLOB

Description: Contains the sequence number of the site object.

2.3.3.9 PROPID_S_FULL_NAME

Value: 309

Variant type: VT_LPWSTR

Description: Contains the DN of the site in Active Directory. (The name format is specified in [MS-ADTS].)

Note Not valid for MSMQ 1.0.

2.3.3.10 PROPID_S_NT4_STUB

Value: 310

Variant type: VT_UI2

Description: Specifies whether the site was migrated from an MQIS database. The value MUST be one of the following.

Value	Description
0x01	Site was migrated from MQIS.
0x00	Site was not migrated.

Note Not valid for MSMQ 1.0.

2.3.3.11 PROPID_S_FOREIGN

Value: 311

Variant type: VT_UI1

Description: Specifies whether the site is used as a definition of an external messaging system. The value MUST be one of the following.

Value	Description
0x01	Site is an external system.
0x00	Site is not external.

Note Not valid for MSMQ 1.0.

2.3.3.12 PROPID_S_DONOTHING

Value: 312

Variant type: VT_UI1

Description: When operating in MSMQ mixed-mode, the MSMQ replication service uses this property when replicating objects from MQIS to Active Directory. Certain properties that are no longer used in MSMQ 2.0 and later are mapped to this property during replication, prior to creating the object in Active Directory. The value stored in this property is not used by MSMQ.

Note Not valid for MSMQ 1.0.

2.3.3.13 PROPID_S_SECURITY

Value: 1301

Variant type: VT_BLOB

Description: Contains the security descriptor of the site object. The BLOB layout is that of SECURITY_DESCRIPTOR, as specified in [MS-DTYP] section 2.4.6.<33>

2.3.3.14 PROPID_S_PSC_SIGNPK

Value: 1302

Variant type: VT_BLOB

Description: Contains the signing key of the PSC formatted as an MQDSPUBLICKEYS (section 2.2.2) structure.

2.3.3.15 PROPID_S_SECURITY_INFORMATION

Value: 1303

Variant type: VT_UI4

Description: The SECURITY_INFORMATION (section 2.2.3) associated with setting or retrieving a security descriptor.

Note Not valid for MSMQ 1.0 or MSMQ 2.0.

2.3.4 Connected Network Property Identifiers

Connected Network object properties contain attributes of a connected network.

2.3.4.1 PROPID_CN_PROTOCOLID

Value: 501

Variant type: VT_UI1

Description: Indicates what network protocol is used on the connected network. This property MUST have one of the values listed in the following table.

Value	Meaning
IP_ADDRESS_TYPE 0x01	The connected network uses IP.
IPX_ADDRESS_TYPE 0x03	The connected network uses IPX.
FOREIGN_ADDRESS_TYPE 0x05	The connected network uses any other network protocol.

2.3.4.2 PROPID_CN_NAME

Value: 502

Variant type: VT_LPWSTR

Description: User-defined name for the connected network, formatted as a null-terminated Unicode string.

2.3.4.3 PROPID_CN_GUID

Value: 503

Variant type: VT_CLSID

Description: A GUID that uniquely identifies the connected network.

2.3.4.4 PROPID_CN_PARTITIONID

Value: 504

Variant type: VT_CLSID

Description: This property MAY<34> be used to group MSMQ directory objects.

2.3.4.5 PROPID_CN_SEQNUM

Value: 505

Variant type: VT_BLOB

Description: Contains the sequence number of the connected network object.

2.3.4.6 PROPID_CN_SECURITY

Value: 1501

Variant type: VT_BLOB

Description: Contains the security descriptor of the connected network object. The BLOB layout is that of SECURITY_DESCRIPTOR, as specified in [MS-DTYP] section 2.4.6.<35>

2.3.5 Enterprise Object Property Identifiers

Enterprise object properties pertain to enterprise-wide settings.

2.3.5.1 PROPID_E_NAME

Value: 601

Variant type: VT_LPWSTR

Description: User-defined name for the enterprise, formatted as a null-terminated Unicode string.

2.3.5.2 PROPID_E_NAMESTYLE

Value: 602

Variant type: VT_UI1

Description: In MSMQ 1.0, this property is not used. In MSMQ 2.0 and up, this property indicates whether weakened security is enabled. Value MUST be one of the following.

Value	Description
0x00	Weakened security is not enabled.
0x01	Weakened security is enabled.
0x02	Use internal default.<36>

2.3.5.3 PROPID_E_CSP_NAME

Value: 603

Variant type: VT_LPWSTR

Description: The type of cryptographic provider used by MSMQ. The default value is "Microsoft Enhanced RSA and AES Cryptographic Provider".

2.3.5.4 PROPID_E_PECNAME

Value: 604

Variant type: VT_LPWSTR

Description: Contains the machine name of the Primary Enterprise Controller, formatted as a null-terminated Unicode string.

2.3.5.5 PROPID_E_S_INTERVAL1

Value: 605

Variant type: VT_UI2

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.5.6 PROPID_E_S_INTERVAL2

Value: 606

Variant type: VT_UI2

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.5.7 PROPID_E_PARTITIONID

Value: 607

Variant type: VT_CLSID

Description: This property MAY<37> be used to group MSMQ directory objects.

2.3.5.8 PROPID_E_SEQNUM

Value: 608

Variant type: VT_BLOB

Description: Contains the sequence number of the enterprise object.

2.3.5.9 PROPID_E_ID

Value: 609

Variant type: VT_CLSID

Description: The GUID identifier for the directory object instance.

2.3.5.10 PROPID_E_CRL

Value: 610

Variant type: VT_BLOB

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.5.11 PROPID_E_CSP_TYPE

Value: 611

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.5.12 PROPID_E_ENCRYPT_ALG

Value: 612

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.5.13 PROPID_E_SIGN_ALG

Value: 613

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.5.14 PROPID_E_HASH_ALG

Value: 614

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.5.15 PROPID_E_LONG_LIVE

Value: 616

Variant type: VT_UI4

Description: The default value for the time, in seconds, that a message has to reach a queue when sending MSMQ messages.

2.3.5.16 PROPID_E_VERSION

Value: 617

Variant type: VT_UI2

Description: The version number of MSMQ Directory Service (MQDS) information.

2.3.5.17 PROPID_E_SECURITY

Value: 1601

Variant type: VT_BLOB

Description: Contains the security descriptor of the enterprise object. The BLOB layout is that of SECURITY_DESCRIPTOR, as specified in [MS-DTYP] section 2.4.6.<38>

2.3.5.18 PROPID_E_CIPHER_MODE

Value: 615

Variant type: VT_UI4

Description: Reserved. The property value associated with this property identifier is undefined and MUST NOT be interpreted by any protocol implementation.

2.3.6 User Object Property Identifiers

User object properties are used by MSMQ during management of user certificates that are stored in the MSMQ Directory Service.

2.3.6.1 PROPID_U_SID

Value: 701

Variant type: VT_BLOB

Description: Contains the user's SID. The **SID** layout is specified in [MS-DTYP] section 2.4.2.2. The **SubAuthority** field of the **SID** packet is a variable-length array of unsigned 32-bit little-endian integers.

2.3.6.2 PROPID_U_PARTITIONID

Value: 703

Variant type: VT_CLSID

Description: This property MAY<39> be used to group MSMQ directory objects.

2.3.6.3 PROPID_U_SEQNUM

Value: 704

Variant type: VT_BLOB

Description: Contains the sequence number of the user object.

2.3.6.4 PROPID_U_SIGN_CERT

Value: 702

Variant type: VT_BLOB

Description: Contains an MQUSERSIGNCERTS structure that packs multiple X.509 encoded certificates for the user object.

2.3.6.5 PROPID_U_DIGEST

Value: 705

Variant type: VT_CLSID | VT_VECTOR

Description: Contains an array of certificate digests. Each digest is computed as the MD5 hash of the encoded certificate. Each array element MUST contain the 16-byte output of the MD5 algorithm, as specified in [RFC1321].

2.3.6.6 PROPID_U_ID

Value: 706

Variant type: VT_CLSID

Description: The GUID identifying the user object.

2.3.7 Routinglink Property Identifiers

Routinglink properties define the cost of routing a message from one site to another.

2.3.7.1 PROPID_L_NEIGHBOR1

Value: 801

Variant type: VT_CLSID

Description: Contains the GUID of one of the routing sites.

2.3.7.2 PROPID_L_NEIGHBOR2

Value: 802

Variant type: VT_CLSID

Description: Contains the GUID of the other routing site.

2.3.7.3 PROPID_L_COST

Value: 803

Variant type: VT_UI4

Description: Contains the cost of the link. Each routing link is assigned a relative cost, which reflects the speed or the monetary cost of the underlying physical communication link. The default value is 1; and costs can range from 1 to 999999, inclusive.

2.3.7.4 PROPID_L_PARTITIONID

Value: 804

Variant type: VT_CLSID

Description: This property MAY<40> be used to group MSMQ directory objects.

2.3.7.5 PROPID_L_SEQNUM

Value: 805

Variant type: VT_BLOB

Description: Contains the sequence number of the routing link object.

2.3.7.6 PROPID_L_ID

Value: 806

Variant type: VT_CLSID

Description: Contains the GUID of the routing link object.

2.3.7.7 PROPID_L_GATES_DN

Value: 807

Variant type: VT_LPWSTR | VT_VECTOR

Description: Contains the distinguished names of the MSMQ Configuration object of the computers that are site gates on the link. Each name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

Note Not valid for MSMQ 1.0 or MSMQ 2.0.

2.3.7.8 PROPID_L_NEIGHBOR1_DN

Value: 808

Variant type: VT_LPWSTR

Description: Contains the distinguished name of one site on the link. The name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

Note Not valid for MSMQ 1.0.

2.3.7.9 PROPID_L_NEIGHBOR2_DN

Value: 809

Variant type: VT_LPWSTR

Description: Contains the distinguished name of the other site on the link. The name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

Note Not valid for MSMQ 1.0.

2.3.7.10 PROPID_L_DESCRIPTION

Value: 810

Variant type: VT_LPWSTR

Description: Contains the description of the routing link.

Note Not valid for MSMQ 1.0.

2.3.7.11 PROPID_L_FULL_PATH

Value: 811

Variant type: VT_LPWSTR

Description: Contains the distinguished name of the routing link object in the Active Directory. The name MUST conform to ABNF: distinguishedName, as described in [RFC4514].

Note Not valid for MSMQ 1.0.

2.3.7.12 PROPID_L_ACTUAL_COST

Value: 812

Variant type: VT_UI4

Description: Contains the untranslated link cost. The value MUST be in the range from 1 to 999999, inclusive.

Note Not valid for MSMQ 1.0.

2.3.7.13 PROPID_L_GATES

Value: 813

Variant type: VT_CLSID | VT_VECTOR

Description: Contains the GUIDs of the MSMQ Configuration objects of the computers that are site gates on the link.

Note Not valid for MSMQ 1.0.

2.3.8 Settings Property Identifiers

Setting objects represent MSMQ Routing Servers or MSMQ Directory Service Servers.

Note All Settings properties are not valid for MSMQ 1.0.

2.3.8.1 PROPID_SET_NAME

Value: 5101

Variant type: VT_LPWSTR

Description: Contains the **Common-Name** attribute, which MUST always be set to the string "MSMQ Settings".

2.3.8.2 PROPID_SET_SERVICE

Value: 5102

Variant type: VT_UI4

Description: Contains a value that identifies the type of service. The value MUST be one of the following.

Value	Description
0x00000000	None
0x00000001	Routing server (SRV)
0x00000002	Backup Site Controller (BSC)
0x00000004	Primary Site Controller (PSC)
0x00000008	Primary Enterprise Controller (PEC)

2.3.8.3 PROPID_SET_QM_ID

Value: 5103

Variant type: VT_CLSID

Description: Contains the GUID of the computer's MSMQ Configuration object.

2.3.8.4 PROPID_SET_FULL_PATH

Value: 5105

Variant type: VT_LPWSTR

Description: Contains the distinguished name of the MSMQ Settings object in the Active Directory. The name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

2.3.8.5 PROPID_SET_NT4

Value: 5106

Variant type: VT_UI1

Description: Specifies whether the server is MSMQ 1.0.

The value MUST be one of the following.

Value	Description
0x01	Server is MSMQ 1.0.
0x00	Server is not MSMQ 1.0.

2.3.8.6 PROPID_SET_PARTITIONID

Value: 5107

Variant type: VT_CLSID

Description: This property MAY<41> be used to group MSMQ directory objects.

2.3.8.7 PROPID_SET_SITENAME

Value: 5108

Variant type: VT_LPWSTR

Description: Contains the site name.

2.3.8.8 PROPID_SET_SERVICE_ROUTING

Value: 5109

Variant type: VT_UI1

Description: Specifies whether the server is a routing server. The value MUST be one of the following.

Value	Description
0x01	Server is a routing server.
0x00	Server is not a routing server.

2.3.8.9 PROPID_SET_SERVICE_DSSERVER

Value: 5110

Variant type: VT_UI1

Description: Specifies whether the queue manager provides access to the Active Directory for MSMQ 2.0 directory service clients. The value MUST be set to one of the following.

Value	Description
0x01	Server provides access to Active Directory.
0x00	Server does not provide access to Active Directory.

2.3.8.10 PROPID_SET_SERVICE_DEPCLIENTS

Value: 5111

Variant type: VT_UI1

Description: Specifies whether the server can be a supporting server for applications. The value MUST be set to one of the following.

Value	Description
0x01	Server can be a supporting server.
0x00	Server cannot be a supporting server.

2.3.8.11 PROPID_SET_OLDSERVICE

Value: 5112

Variant type: VT_UI4

Description: Contains a value that identifies the type of service. The value MUST be set to one of the following.

Value	Description
0x00000000	None
0x00000001	Routing server (SRV)
0x00000002	Backup Site Controller (BSC)
0x00000004	Primary Site Controller (PSC)
0x00000008	Primary Enterprise Controller (PEC)

2.3.9 MQUser Property Identifiers

These properties represent attributes of users who migrated to Active Directory from the Microsoft MQIS.

Note These values are not valid for MSMQ 1.0.

2.3.9.1 PROPID_MQU_SID

Value: 5401

Variant type: VT_BLOB

Description: The migrated user's SID. The **SID** layout is specified in [MS-DTYP] section 2.4.2.2. The **SubAuthority** field of the **SID** packet is a variable-length array of unsigned 32-bit little-endian integers.

2.3.9.2 PROPID_MQU_SIGN_CERT

Value: 5402

Variant type: VT_BLOB

Description: Contains an X.509 encoded certificate for the migrated user object as specified in [RFC3280].

2.3.9.3 PROPID_MQU_DIGEST

Value: 5405

Variant type: VT_CLSID | VT_VECTOR

Description: Contains an array of certificate digests. Each digest is computed as the MD5 hash of the encoded certificate. Each array element MUST contain the 16-byte output of the MD5 algorithm, as specified in [RFC1321].

2.3.9.4 PROPID_MQU_ID

Value: 5406

Variant type: VT_CLSID

Description: Contains the GUID of the MQUser object.

2.3.9.5 PROPID_MQU_SECURITY

Value: 5407

Variant type: VT_BLOB

Description: Contains the security descriptor of the MQUser object. The BLOB layout is that of **SECURITY_DESCRIPTOR**, as specified in [MS-DTYP] section 2.4.6.<42>

2.3.10 Computer Property Identifiers

Computer properties contain attributes of the computer object.

Note These values are not valid for MSMQ 1.0.

2.3.10.1 PROPID_COM_FULL_PATH

Value: 5201

Variant type: VT_LPWSTR

Description: Contains the distinguished name of the computer. The name MUST conform to ABNF: distinguishedName, as specified in [RFC4514].

2.3.10.2 PROPID_COM_SAM_ACCOUNT

Value: 5202

Variant type: VT_LPWSTR

Description: Identifies a property that contains the name of the computer account in Active Directory. Contains the name of the computer object. The value is represented as the computer name (truncated to 19 characters) followed by a dollar sign "$" character. For example, the property value for a computer with the name "MyComputer" is "MyComputer$".

2.3.10.3 PROPID_COM_ACCOUNT_CONTROL

Value: 5204

Variant type: VT_UI4

Description: Contains user account control attributes, as specified in [MS-SAMR]. The value MUST be a bitmask computed as a logical OR of a set of UF_FLAG codes, as specified in [MS-SAMR] section 2.2.1.13.

2.3.10.4 PROPID_COM_DNS_HOSTNAME

Value: 5205

Variant type: VT_LPWSTR

Description: Contains the DNS host name attribute of the computer object. The value MUST contain the FQDN of the computer.

2.3.10.5 PROPID_COM_SID

Value: 5206

Variant type: VT_BLOB

Description: Contains the SID of the computer object. This property is read from Active Directory during creation of an MSMQ service Configuration object, and is used to add the computer SID to the MSMQ service configuration DACL. The **SID** layout is specified in [MS-DTYP] section 2.4.2.2. The **SubAuthority** field of the **SID** packet is a variable-length array of unsigned 32-bit little-endian integers.

2.3.10.6 PROPID_COM_SIGN_CERT

Value: 5207

Variant type: VT_BLOB

Description: Contains an X.509 encoded certificate for the computer object. The X.509 encoded certificate is specified in [RFC3280].

2.3.10.7 PROPID_COM_DIGEST

Value: 5208

Variant type: VT_CLSID | VT_VECTOR

Description: Contains an array of certificate digests. The digest is computed as the MD5 hash of the encoded certificate. Each value MUST contain the 16-byte output of the MD5 algorithm, as specified in [RFC1321].

2.3.10.8 PROPID_COM_ID

Value: 5209

Variant type: VT_CLSID

Description: Contains the GUID (as specified in [MS-DTYP] section 2.3.4) of the computer object.

2.3.11 Management Machine Property Identifiers

Management machine property identifiers provide values that identify properties that describe local administration of MSMQ machines.

2.3.11.1 PROPID_MGMT_MSMQ_ACTIVEQUEUES

Value: 1

Variant type: VT_LPWSTR | VT_VECTOR

Description: A list of all the active queue names on the computer. Each name MUST conform to the ABNF for a format name, as specified in sections 2.1.2, 2.1.3, 2.1.4, and 2.1.6.

2.3.11.2 PROPID_MGMT_MSMQ_PRIVATEQ

Value: 2

Variant type: VT_LPWSTR | VT_VECTOR

Description: A list of the path names of all the private queues registered on the computer.

2.3.11.3 PROPID_MGMT_MSMQ_DSSERVER

Value: 3

Variant type: VT_LPWSTR

Description: The name of the current MSMQ Directory Service server for the computer. The pointer to a null-terminated Unicode string that specifies the computer name of the discovered server. The returned computer name is prefixed with "\\".

The format in ABNF notation is as follows.

```
DS = "\\" 1*NameChar EndList
```

```
R1 = %x01-2b ; Range 1
R2 = %x2c-2c ; Range 2 is x2c only
R3 = %x00-ff ; Range 3
R4 = %x01-ff ; Range 4
R5 = %x2d-ff ; Range 5
R6 = %x00-00 ; Range 6 is x00 only
X1 = R1 R3 ; Two hex digit range 1
X2 = R2 R4 ; Two hex digit range 2
X3 = R5 R3 ; Two hex digit range 3
X4 = R6 R4 ; Two hex digit range 4
NameChar = X1 / X2 / X3 / X4 ; Name character: no commas or nulls
EndList =  %x00.00 ; Use null for end of string
```

2.3.11.4 PROPID_MGMT_MSMQ_CONNECTED

Value: 4

Variant type: VT_LPWSTR

Description: The value that indicates whether the queue manager on the computer has been disconnected from the network. The value MUST be one of the following strings.

Value	Constant
"CONNECTED"	MSMQ_CONNECTED
"DISCONNECTED"	MSMQ_DISCONNECTED

2.3.11.5 PROPID_MGMT_MSMQ_TYPE

Value: 5

Variant type: VT_LPWSTR

Description: The version and build information for the computer operating system and MSMQ installation.

2.3.11.6 PROPID_MGMT_MSMQ_BYTES_IN_ALL_QUEUES

Value: 6

Variant type: VT_I8

Description: The number of message bytes stored in all the queues on the computer.

Note Not valid for MSMQ 1.0 and MSMQ 2.0.

2.3.12 Management Queue Property Identifiers

Management queue property identifiers provide values that identify properties for monitoring the MSMQ installation and the queues on a computer, which allows applications to manage these resources programmatically.

2.3.12.1 PROPID_MGMT_QUEUE_PATHNAME

Value: 1

Variant type: VT_LPWSTR

Description: The path name of the queue. The path name format is specified in section 2.1.1.

2.3.12.2 PROPID_MGMT_QUEUE_FORMATNAME

Value: 2

Variant type: VT_LPWSTR

Description: The format name of the queue, as specified in section 2.1.

2.3.12.3 PROPID_MGMT_QUEUE_TYPE

Value: 3

Variant type: VT_LPWSTR

Description: The type of the queue. The value MUST be one of the following strings.

Value	Constant
"PUBLIC"	MGMT_QUEUE_TYPE_PUBLIC
"PRIVATE"	MGMT_QUEUE_TYPE_PRIVATE
"MACHINE"	MGMT_QUEUE_TYPE_MACHINE
"CONNECTOR"	MGMT_QUEUE_TYPE_CONNECTOR
"MULTICAST"	MGMT_QUEUE_TYPE_MULTICAST

2.3.12.4 PROPID_MGMT_QUEUE_LOCATION

Value: 4

Variant type: VT_LPWSTR

Description: The value that indicates whether the queue is located on the computer. The value MUST be one of the following strings.

Value	Constant
"LOCAL"	MGMT_QUEUE_LOCAL_LOCATION
"REMOTE"	MGMT_QUEUE_REMOTE_LOCATION

2.3.12.5 PROPID_MGMT_QUEUE_XACT

Value: 5

Variant type: VT_LPWSTR

Description: The value that indicates whether the queue is transactional. The value MUST be one of the following strings.

Value	Constant
"UNKNOWN"	MGMT_QUEUE_UNKNOWN_TYPE
"YES"	MGMT_QUEUE_TRANSACTIONAL_TYPE
"NO"	MGMT_QUEUE_NOT_TRANSACTIONAL_TYPE

2.3.12.6 PROPID_MGMT_QUEUE_FOREIGN

Value: 6

Variant type: VT_LPWSTR

Description: The string that indicates whether the queue is a foreign queue. The value MUST be one of the following strings.

Value	Constant
"UNKNOWN"	MGMT_QUEUE_UNKNOWN_TYPE
"YES"	MGMT_QUEUE_FOREIGN_TYPE
"NO"	MGMT_QUEUE_NOT_FOREIGN_TYPE

2.3.12.7 PROPID_MGMT_QUEUE_MESSAGE_COUNT

Value: 7

Variant type: VT_UI4

Description: The number of messages in the queue.

2.3.12.8 PROPID_MGMT_QUEUE_BYTES_IN_QUEUE

Value: 8

Variant type: VT_UI4

Description: The number of message bytes for all messages in the queue.

Note This property identifier is available only in MSMQ 3.0 and later versions. It replaces PROPID_MGMT_QUEUE_JOURNAL_USED_QUOTA from MSMQ 1.0 and MSMQ 2.0.

2.3.12.9 PROPID_MGMT_QUEUE_JOURNAL_MESSAGE_COUNT

Value: 9

Variant type: VT_UI4

Description: The number of messages in the queue journal.

2.3.12.10 PROPID_MGMT_QUEUE_BYTES_IN_JOURNAL

Value: 10

Variant type: VT_UI4

Description: The number of message bytes for all messages in the queue journal.

Note This property identifier is available only in MSMQ 3.0 and later versions. It replaces PROPID_MGMT_QUEUE_JOURNAL_USED_QUOTA from MSMQ 1.0 and MSMQ 2.0.

2.3.12.11 PROPID_MGMT_QUEUE_STATE

Value: 11

Variant type: VT_LPWSTR

Description: The connection state of the outgoing queue. The value MUST be one of the following strings.

Value	Constant
"LOCAL CONNECTION"	MGMT_QUEUE_STATE_LOCAL
"INACTIVE"	MGMT_QUEUE_STATE_NONACTIVE
"WAITING"	MGMT_QUEUE_STATE_WAITING
"NEED VALIDATION"	MGMT_QUEUE_STATE_NEED_VALIDATE
"ONHOLD"	MGMT_QUEUE_STATE_ONHOLD
"CONNECTED"	MGMT_QUEUE_STATE_CONNECTED
"DISCONNECTING"	MGMT_QUEUE_STATE_DISCONNECTING
"DISCONNECTED"	MGMT_QUEUE_STATE_DISCONNECTED

2.3.12.12 PROPID_MGMT_QUEUE_NEXTHOPS

Value: 12

Variant type: VT_LPWSTR | VT_VECTOR

Description: The address, or a list of possible addresses, for routing messages to the destination queue in the next hop. If the queue is in the process of being connected, a list of possible addresses is returned. Each element conforms to the following ABNF.

```
Address = IPAddr/ForeignAddress/IPv6Addr

IPAddr    = "IP=" IPv4address

ForeignAddress = "FOREIGN=" Guid

IPv6Addr    = "IPv6=" IPv6address

Guid = 8HexDig %x2D 3(4HexDig %x2D) 12HexDig

HexDig = Digit / "A" /  "B" / "C" / "D"  / "E" / "F"
```

```
Digit = %x30-39
```

ABNF rules IPv4address and IPv6address are defined in [RFC3986] appendix A.

2.3.12.13 PROPID_MGMT_QUEUE_EOD_LAST_ACK

Value: 13

Variant type: VT_BLOB

Description: The sequence information about the last message sent from the computer to the queue for which an order acknowledgment was received. The BLOB layout of the SEQUENCE_INFO structure is specified in section 2.2.5.

2.3.12.14 PROPID_MGMT_QUEUE_EOD_LAST_ACK_TIME

Value: 14

Variant type: VT_I4

Description: The date and time when the last order acknowledgment for a message sent from the computer to the queue was received. Time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

2.3.12.15 PROPID_MGMT_QUEUE_EOD_LAST_ACK_COUNT

Value: 15

Variant type: VT_UI4

Description: The number of times that the last order acknowledgment for a message sent from the computer to the queue was received.

2.3.12.16 PROPID_MGMT_QUEUE_EOD_FIRST_NON_ACK

Value: 16

Variant type: VT_BLOB

Description: The sequence information about the first message sent from the computer to the queue for which no order acknowledgment was received. The BLOB layout of the SEQUENCE_INFO structure is specified in section 2.2.5.

2.3.12.17 PROPID_MGMT_QUEUE_EOD_LAST_NON_ACK

Value: 17

Variant type: VT_BLOB

Description: The sequence information about the last message that was sent from the computer to the queue for which no order acknowledgment was received. The BLOB layout of the SEQUENCE_INFO structure is specified in section 2.2.5.

2.3.12.18 PROPID_MGMT_QUEUE_EOD_NEXT_SEQ

Value: 18

Variant type: VT_BLOB

Description: The sequence information about the next message to be sent from the computer to the queue. The BLOB layout of the SEQUENCE_INFO structure is specified in section 2.2.5.

2.3.12.19 PROPID_MGMT_QUEUE_EOD_NO_READ_COUNT

Value: 19

Variant type: VT_UI4

Description: The number of messages sent from the computer to the queue for which an order acknowledgment was received but for which a receive acknowledgment message was not received.

2.3.12.20 PROPID_MGMT_QUEUE_EOD_NO_ACK_COUNT

Value: 20

Variant type: VT_UI4

Description: The number of messages sent from the computer to the queue for which no order acknowledgment was received.

2.3.12.21 PROPID_MGMT_QUEUE_EOD_RESEND_TIME

Value: 21

Variant type: VT_I4

Description: The time at which MSMQ will attempt to send a message from the computer to the queue again. Time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

2.3.12.22 PROPID_MGMT_QUEUE_EOD_RESEND_INTERVAL

Value: 22

Variant type: VT_UI4

Description: The resend interval (in seconds) for the messages in the outgoing queue for which no order acknowledgment was received.

2.3.12.23 PROPID_MGMT_QUEUE_EOD_RESEND_COUNT

Value: 23

Variant type: VT_UI4

Description: The number of times that the last message in the corresponding outgoing queue on the computer was sent.

2.3.12.24 PROPID_MGMT_QUEUE_EOD_SOURCE_INFO

Value: 24

Variant type: VT_VARIANT | VT_VECTOR

Description: The array of information about the transactional messages sent from all source computers to the queue on the target computer.

The array contains the following six items. Each item is an array; there is one entry in each array for each message.

- **Format name**

  Variant type: VT_LPWSTR | VT_VECTOR

  Description: Each entry is a format name of a queue, as specified in section 2.1.

- **Sender ID**

  Variant type: VT_CLSID | VT_VECTOR

  Description: Each entry is a GUID of the sender of the message.

- **Sequence ID**

  Variant type: VT_UI8 | VT_VECTOR

  Description: Each entry is a number to distinguish a sequence from other sequences.

- **Sequence number**

  Variant type: VT_UI4 | VT_VECTOR

  Description: Each entry is a sequence number.

- **Last access time**

  Variant type: VT_I4 | VT_VECTOR

  Description: Each entry was the time when the queue was accessed. Time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

- **Message reject count**

  Variant type: VT_UI4 | VT_VECTOR

  Description: Each entry is the number of times that a message was rejected.

2.3.12.25 PROPID_MGMT_QUEUE_CONNECTION_HISTORY

Value: 25

Variant type: 12 | 0x1000 - VT_VARIANT | VT_VECTOR

Description: The array representing the queue connection state history information. The array consists of the following four items, and each item is an array.

- **Status**

  Variant type: VT_UI4 | VT_VECTOR

  Description: Each entry is the connection status and cause of the failure. The contents MUST be one of the following values.

Value	Description
0x00000000	Connection is in the process of establishment; no failures have occurred.
0x00000001	Connection establishment packet has been received.
0x00000002	Connection has been successfully established and is ready to send messages.
0x80000000	Exact reason for failure cannot be determined.
0x80000001	Ping failure.
0x80000002	Create socket failure.
0x80000003	Bind socket failure.
0x80000004	Connect socket failure.
0x80000005	TCP is not enabled.
0x80000006	Send operation on a socket failed.
0x80000007	Send operation failed because connection is not ready.
0x80000008	DNS failure.
0x80000009	Could not validate server certificate in HTTPS scenario.
0x8000000A	Connection limit reached, cannot establish new session to a specific destination.
0x8000000B	Connection refused by other side due to any reason (quota, invalid packet, connection limit reached).
0x8000000C	Absence of MSMQ Directory Service server connectivity prevents getting routing data.
0x8000000D	Failure due to low memory.

- **Time at which the failure occurred**

  Variant type: VT_I4 | VT_VECTOR

  Description: Each entry is the time is represented as the number of seconds elapsed since midnight (00:00:00), January 1, 1970 UTC.

- **Error indicator**

  Variant type: VT_I4 | VT_VECTOR

  Description: Each entry is a value; a non-zero value indicates an error.

- **List of addresses**

  Variant type: VT_LPWSTR | VT_VECTOR | VT_VECTOR

  Description: Each entry is an address or a list of possible addresses for routing messages to the destination queue in the next hop. Each element MUST conform to the following ABNF.

```
Address = Name / IPaddress

Name = "\\" 1*Alpha

Alpha = %x41-5A / %x61-7A

IPaddress = AddDigits  3("." AddDigits)
```

```
Digit = %x30-39

AddDigits = 1*3Digit
```

2.3.12.26 PROPID_MGMT_QUEUE_SUBQUEUE_COUNT

Value: 26

Variant type: VT_UI4

Description: The count of the number of subqueues for a given queue.

2.3.12.27 PROPID_MGMT_QUEUE_SUBQUEUE_NAMES

Value: 27

Variant type: VT_LPWSTR | VT_VECTOR

Description: The list of subqueue names (as specified in section 2.1) for a given queue.

2.3.13 Deletion Notification Property Identifiers

Deletion notification property identifiers provide values that identify properties in a directory service object deletion notification.

2.3.13.1 PROPID_D_SEQNUM

Value: 1401

Variant type: VT_BLOB

Description: Contains the sequence number of the deleted object.

2.3.13.2 PROPID_D_PARTITIONID

Value: 1402

Variant type: VT_CLSID

Description: This property MAY<43> be used to group MSMQ directory objects.

2.3.13.3 PROPID_D_SCOPE

Value: 1403.

Variant type: VT_UI1.

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

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

2.3.13.4 PROPID_D_OBJTYPE

Value: 1404.

Variant type: VT_UI1.

Description: A value that specifies the type of a deleted object. The value 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_ROUTINGLINK	Object represents a routing link.

2.3.13.5 PROPID_D_IDENTIFIER

Value: 1405

Variant type: VT_CLSID

Description: Contains the GUID of the deleted object.

2.4 Error Codes

The following table specifies MSMQ-specific HRESULT values. Not all methods of all protocols return these error codes. Common **HRESULT** values are specified in [MS-ERREF] section 2.1.<44>

Return value/code	Description
0x400E0001 MQ_INFORMATION_PROPERTY	One or more of the properties passed resulted in a warning, but the function completed.
0x400E0002 MQ_INFORMATION_ILLEGAL_PROPERTY	The property ID is invalid.
0x400E0003 MQ_INFORMATION_PROPERTY_IGNORED	The property specified was ignored for this operation.
0x400E0004 MQ_INFORMATION_UNSUPPORTED_PROPERTY	The property specified is not supported and was ignored for this operation.

Return value/code	Description
0x400E0005 MQ_INFORMATION_DUPLICATE_PROPERTY	The property specified is already in the property identifier array. The duplicate was ignored for this operation.
0x400E0006 MQ_INFORMATION_OPERATION_PENDING	An asynchronous operation is currently pending.
0x400E0009 MQ_INFORMATION_FORMATNAME_BUFFER_TOO_SMALL	The format name buffer supplied was too small to hold the format name; however, the queue was created successfully.
0x400E000A MQ_INFORMATION_INTERNAL_USER_CERT_EXIST	An internal Message Queuing certificate already exists for this user.
0x400E000B MQ_INFORMATION_OWNER_IGNORED	The queue owner was not set during the processing of this call.
0xC00E0001 MQ_ERROR	Generic error code.
0xC00E0002 MQ_ERROR_PROPERTY	One or more of the properties passed are invalid.
0xC00E0003 MQ_ERROR_QUEUE_NOT_FOUND	The queue does not exist or you do not have sufficient permissions to perform the operation.
0xC00E0004 MQ_ERROR_QUEUE_NOT_ACTIVE	The queue is not open or might not exist.
0xC00E0005 MQ_ERROR_QUEUE_EXISTS	A queue with the same path name already exists.
0xC00E0006 MQ_ERROR_INVALID_PARAMETER	An invalid parameter was passed to a function.
0xC00E0007 MQ_ERROR_INVALID_HANDLE	An invalid handle was passed to a function.
0xC00E0008 MQ_ERROR_OPERATION_CANCELLED	The operation was canceled before it could be completed.
0xC00E0009 MQ_ERROR_SHARING_VIOLATION	There is a sharing violation. The queue is already open for exclusive retrieval.
0xC00E000B MQ_ERROR_SERVICE_NOT_AVAILABLE	The Message Queuing service is not available.
0xC00E000D MQ_ERROR_MACHINE_NOT_FOUND	The computer specified cannot be found.
0xC00E0010 MQ_ERROR_ILLEGAL_SORT	The sort operation specified is invalid; for example, there are duplicate columns.
0xC00E0011 MQ_ERROR_ILLEGAL_USER	The user specified is not a valid user.
0xC00E0013	A connection with Active Directory cannot

Return value/code	Description
MQ_ERROR_NO_DS	be established. Verify that there are sufficient permissions to perform this operation.
0xC00E0014 MQ_ERROR_ILLEGAL_QUEUE_PATHNAME	The queue path name specified is invalid.
0xC00E0018 MQ_ERROR_ILLEGAL_PROPERTY_VALUE	The property value specified is invalid.
0xC00E0019 MQ_ERROR_ILLEGAL_PROPERTY_VT	The VARTYPE value specified is invalid.
0xC00E001A MQ_ERROR_BUFFER_OVERFLOW	The buffer supplied for message property retrieval is too small. The message was not removed from the queue, but the part of the message property that was in the buffer was copied.
0xC00E001B MQ_ERROR_IO_TIMEOUT	The time specified to wait for the message elapsed.
0xC00E001C MQ_ERROR_ILLEGAL_CURSOR_ACTION	The MQ_ACTION_PEEK_NEXT value specified cannot be used with the current cursor position.
0xC00E001D MQ_ERROR_MESSAGE_ALREADY_RECEIVED	The message to which the cursor is currently pointing was removed from the queue by another process or by another call without the use of this cursor.
0xC00E001E MQ_ERROR_ILLEGAL_FORMATNAME	The format name specified is invalid.
0xC00E001F MQ_ERROR_FORMATNAME_BUFFER_TOO_SMALL	The format name buffer supplied to the API was too small to hold the format name.
0xC00E0020 MQ_ERROR_UNSUPPORTED_FORMATNAME_OPERATION	Operations of the type requested (for example, deleting a queue using a direct format name) are not supported for the format name specified.
0xC00E0021 MQ_ERROR_ILLEGAL_SECURITY_DESCRIPTOR	The specified security descriptor is invalid.
0xC00E0022 MQ_ERROR_SENDERID_BUFFER_TOO_SMALL	The size of the buffer for the user ID property is too small.
0xC00E0023 MQ_ERROR_SECURITY_DESCRIPTOR_TOO_SMALL	The size of the buffer passed is too small.
0xC00E0024 MQ_ERROR_CANNOT_IMPERSONATE_CLIENT	The security credentials cannot be verified because the RPC server cannot impersonate the client application.
0xC00E0025 MQ_ERROR_ACCESS_DENIED	Access is denied.
0xC00E0026	The client does not have sufficient

Return value/code	Description
MQ_ERROR_PRIVILEGE_NOT_HELD	security privileges to perform the operation.
0xC00E0027 MQ_ERROR_INSUFFICIENT_RESOURCES	There are insufficient resources to perform this operation.
0xC00E0028 MQ_ERROR_USER_BUFFER_TOO_SMALL	The request failed because the user buffer is too small to hold the information returned.
0xC00E002A MQ_ERROR_MESSAGE_STORAGE_FAILED	A recoverable or journal message could not be stored. The message was not sent.
0xC00E002B MQ_ERROR_SENDER_CERT_BUFFER_TOO_SMALL	The buffer for the user certificate property is too small.
0xC00E002C MQ_ERROR_INVALID_CERTIFICATE	The user certificate is invalid.
0xC00E002D MQ_ERROR_CORRUPTED_INTERNAL_CERTIFICATE	The internal Message Queuing certificate is corrupted.
0xC00E002E MQ_ERROR_INTERNAL_USER_CERT_EXIST	An internal Message Queuing certificate already exists for this user.
0xC00E002F MQ_ERROR_NO_INTERNAL_USER_CERT	No internal Message Queuing certificate exists for the user.
0xC00E0030 MQ_ERROR_CORRUPTED_SECURITY_DATA	A cryptographic function failed.
0xC00E0031 MQ_ERROR_CORRUPTED_PERSONAL_CERT_STORE	The personal certificate store is corrupted.
0xC00E0033 MQ_ERROR_COMPUTER_DOES_NOT_SUPPORT_ENCRYPTION	The computer does not support encryption operations.
0xC00E0035 MQ_ERROR_BAD_SECURITY_CONTEXT	The security context is invalid.
0xC00E0036 MQ_ERROR_COULD_NOT_GET_USER_SID	The SID cannot be obtained from the thread token.
0xC00E0037 MQ_ERROR_COULD_NOT_GET_ACCOUNT_INFO	The account information for the user cannot be obtained.
0xC00E0038 MQ_ERROR_ILLEGAL_MQCOLUMNS	The MQCOLUMNS parameter is invalid.
0xC00E0039 MQ_ERROR_ILLEGAL_PROPID	A property identifier is invalid.
0xC00E003A MQ_ERROR_ILLEGAL_RELATION	A relationship parameter is invalid.
0xC00E003B MQ_ERROR_ILLEGAL_PROPERTY_SIZE	The size of the buffer for the message identifier or correlation identifier is invalid.

Return value/code	Description
0xC00E003C MQ_ERROR_ILLEGAL_RESTRICTION_PROPID	A property identifier specified in MQRESTRICTION is invalid.
0xC00E003D MQ_ERROR_ILLEGAL_MQQUEUEPROPS	Either the pointer to the MQQUEUEPROPS structure has a null value or no properties are specified in it.
0xC00E003E MQ_ERROR_PROPERTY_NOTALLOWED	The property identifier specified is invalid for the operation requested.
0xC00E003F MQ_ERROR_INSUFFICIENT_PROPERTIES	Not all the properties required for the operation were specified for the input parameters.
0xC00E0040 MQ_ERROR_MACHINE_EXISTS	The MSMQ Configuration (msmq) object already exists in Active Directory.
0xC00E0041 MQ_ERROR_ILLEGAL_MQQMPROPS	Either the pointer to the MQQMPROPS structure has a null value, or no properties are specified in it.
0xC00E0042 MQ_ERROR_DS_IS_FULL	Valid for MSMQ 1.0 and MSMQ 2.0. Distinguished name (DN) (DS) is full.
0xC00E0043 MQ_ERROR_DS_ERROR	There is an internal Active Directory error.
0xC00E0044 MQ_ERROR_INVALID_OWNER	The object owner is invalid.
0xC00E0045 MQ_ERROR_UNSUPPORTED_ACCESS_MODE	The access mode specified is unsupported.
0xC00E0046 MQ_ERROR_RESULT_BUFFER_TOO_SMALL	The result buffer specified is too small.
0xC00E0048 MQ_ERROR_DELETE_CN_IN_USE	Valid for MSMQ 1.0 and MSMQ 2.0. The connected network cannot be deleted; it is in use.
0xC00E0049 MQ_ERROR_NO_RESPONSE_FROM_OBJECT_SERVER	There was no response from the object owner.
0xC00E004A MQ_ERROR_OBJECT_SERVER_NOT_AVAILABLE	The object owner is not available.
0xC00E004B MQ_ERROR_QUEUE_NOT_AVAILABLE	An error occurred while reading from a queue located on a remote computer.
0xC00E004C MQ_ERROR_DTC_CONNECT	A connection cannot be established with the Distributed Transaction Coordinator.
0xC00E004E MQ_ERROR_TRANSACTION_IMPORT	The transaction specified cannot be imported.
0xC00E0050 MQ_ERROR_TRANSACTION_USAGE	An attempted action cannot be performed within a transaction.

Return value/code	Description
0xC00E0051 MQ_ERROR_TRANSACTION_SEQUENCE	The transaction's operation sequence is incorrect.
0xC00E0055 MQ_ERROR_MISSING_CONNECTOR_TYPE	The connector type message property is not specified. This property is required for sending an acknowledgment message or a secure message.
0xC00E0056 MQ_ERROR_STALE_HANDLE	The Message Queuing service was restarted. Any open queue handles have to be closed.
0xC00E0058 MQ_ERROR_TRANSACTION_ENLIST	The transaction specified cannot be enlisted.
0xC00E005A MQ_ERROR_QUEUE_DELETED	The queue was deleted. Messages cannot be received anymore using this queue handle. The handle has to be closed.
0xC00E005B MQ_ERROR_ILLEGAL_CONTEXT	The context parameter is invalid.
0xC00E005C MQ_ERROR_ILLEGAL_SORT_PROPID	An invalid property identifier is specified in MQSORTSET.
0xC00E005D MQ_ERROR_LABEL_TOO_LONG	The message label is too long. Its length MUST be less than or equal to MQ_MAX_MSG_LABEL_LEN.
0xC00E005E MQ_ERROR_LABEL_BUFFER_TOO_SMALL	The label buffer supplied to the API is too small.
0xC00E005F MQ_ERROR_MQIS_SERVER_EMPTY	Valid for MSMQ 1.0 and MSMQ 2.0. The list of MQIS servers (in the registry) is empty.
0xC00E0060 MQ_ERROR_MQIS_READONLY_MODE	Valid for MSMQ 1.0 and MSMQ 2.0. The MQIS database is in read-only mode.
0xC00E0061 MQ_ERROR_SYMM_KEY_BUFFER_TOO_SMALL	The buffer passed for the symmetric key is too small.
0xC00E0062 MQ_ERROR_SIGNATURE_BUFFER_TOO_SMALL	The buffer passed for the signature property is too small.
0xC00E0063 MQ_ERROR_PROV_NAME_BUFFER_TOO_SMALL	The buffer passed for the provider name property is too small.
0xC00E0064 MQ_ERROR_ILLEGAL_OPERATION	The operation is invalid for a foreign Message Queuing system.
0xC00E0065 MQ_ERROR_WRITE_NOT_ALLOWED	Another MQIS server is being installed. Write operations to the database are not allowed at this stage.
0xC00E0066 MQ_ERROR_WKS_CANT_SERVE_CLIENT	The MSMQ service cannot be a supporting server. A Message Queuing supporting server is required.
0xC00E0067	The supporting server has reached its

Return value/code	Description
MQ_ERROR_DEPEND_WKS_LICENSE_OVERFLOW	limit for accepting application connections.
0xC00E0068 MQ_CORRUPTED_QUEUE_WAS_DELETED	The corresponding file for the designated queue in the Lqs folder was deleted because it was corrupted.
0xC00E0069 MQ_ERROR_REMOTE_MACHINE_NOT_AVAILABLE	The remote computer is not available.
0xC00E006A MQ_ERROR_UNSUPPORTED_OPERATION	This operation is not supported for Message Queuing installed in workgroup mode.
0xC00E006B MQ_ERROR_ENCRYPTION_PROVIDER_NOT_SUPPORTED	The requested cryptographic service provider is not supported by Message Queuing.
0xC00E006C MQ_ERROR_CANNOT_SET_CRYPTO_SEC_DESCR	The security descriptors for the cryptographic keys cannot be set.
0xC00E006D MQ_ERROR_CERTIFICATE_NOT_PROVIDED	A user attempted to send an authenticated message without a certificate.
0xC00E006E MQ_ERROR_Q_DNS_PROPERTY_NOT_SUPPORTED	The column PROPID_Q_PATHNAME_DNS is not supported for the API.
0xC00E006F MQ_ERROR_CANNOT_CREATE_CERT_STORE	A certificate store cannot be created for the internal certificate.
0xC00E0070 MQ_ERROR_CANNOT_OPEN_CERT_STORE	The certificate store for the internal certificate cannot be opened.
0xC00E0071 MQ_ERROR_ILLEGAL_ENTERPRISE_OPERATION	This operation is invalid for an MsmqServices object.
0xC00E0072 MQ_ERROR_CANNOT_GRANT_ADD_GUID	The Add GUID permission cannot be granted to the current user.
0xC00E0073 MQ_ERROR_CANNOT_LOAD_MSMQOCM	Valid for MSMQ 1.0 and MSMQ 2.0. The dynamic-link library Msmqocm.dll cannot be loaded.
0xC00E0074 MQ_ERROR_NO_ENTRY_POINT_MSMQOCM	An entry point cannot be located in Msmqocm.dll.
0xC00E0075 MQ_ERROR_NO_MSMQ_SERVERS_ON_DC	Message Queuing servers cannot be found on domain controllers.
0xC00E0076 MQ_ERROR_CANNOT_JOIN_DOMAIN	The computer joined the domain, but Message Queuing will continue to run in workgroup mode because it failed to register itself in Active Directory.
0xC00E0077 MQ_ERROR_CANNOT_CREATE_ON_GC	The object was not created on the Global Catalog server specified.
0xC00E0078 MQ_ERROR_GUID_NOT_MATCHING	Valid for MSMQ 1.0 and MSMQ 2.0. Failed to create an msmqConfiguration object

Return value/code	Description
	with a GUID that matches the computer installation. MSMQ must be uninstalled and then reinstalled.
0xC00E0079 MQ_ERROR_PUBLIC_KEY_NOT_FOUND	The public key for the designated computer cannot be found.
0xC00E007A MQ_ERROR_PUBLIC_KEY_DOES_NOT_EXIST	The public key for the designated computer does not exist.
0xC00E007B MQ_ERROR_ILLEGAL_MQPRIVATEPROPS	The parameters in MQPRIVATEPROPS are invalid. Either the pointer to the MQPRIVATEPROPS structure has a null value or no properties are specified in it.
0xC00E007C MQ_ERROR_NO_GC_IN_DOMAIN	Global Catalog servers cannot be found in the domain specified.
0xC00E007D MQ_ERROR_NO_MSMQ_SERVERS_ON_GC	No Message Queuing servers were found on Global Catalog servers.
0xC00E007E MQ_ERROR_CANNOT_GET_DN	Valid for MSMQ 1.0 and MSMQ 2.0. Failed to retrieve the distinguished name (DN) of the local computer.
0xC00E007F MQ_ERROR_CANNOT_HASH_DATA_EX	Data for an authenticated message cannot be hashed.
0xC00E0080 MQ_ERROR_CANNOT_SIGN_DATA_EX	Data cannot be signed before sending an authenticated message.
0xC00E0081 MQ_ERROR_CANNOT_CREATE_HASH_EX	A hash object cannot be created for an authenticated message.
0xC00E0082 MQ_ERROR_FAIL_VERIFY_SIGNATURE_EX	The signature of the message received is not valid.
0xC00E0083 MQ_ERROR_CANNOT_DELETE_PSC_OBJECTS	The delete operation against the designated object failed because the object is owned by a PSC. The operation cannot be performed.
0xC00E0084 MQ_ERROR_NO_MQUSER_OU	There is no MSMQ Users organizational unit object in Active Directory for the domain. Please create one manually.
0xC00E0085 MQ_ERROR_CANNOT_LOAD_MQAD	The dynamic-link library Mqad.dll cannot be loaded.
0xC00E0086 MQ_ERROR_CANNOT_LOAD_MQDSSRV	Obsolete: not used in any version of MSMQ.
0xC00E0087 MQ_ERROR_PROPERTIES_CONFLICT	Two or more of the properties passed cannot coexist.
0xC00E0088 MQ_ERROR_MESSAGE_NOT_FOUND	The message does not exist or was removed from the queue.
0xC00E0089	The sites in which the computer resides cannot be resolved. Verify that the

Return value/code	Description
MQ_ERROR_CANT_RESOLVE_SITES	subnets in the network are configured correctly in Active Directory and that each site is configured with the appropriate subnet.
0xC00E008A MQ_ERROR_NOT_SUPPORTED_BY_DEPENDENT_CLIENTS	This operation is not supported for communicating with a supporting server.
0xC00E008B MQ_ERROR_OPERATION_NOT_SUPPORTED_BY_REMOTE_COMPUTER	This operation is not supported by the remote Message Queuing service.
0xC00E008C MQ_ERROR_NOT_A_CORRECT_OBJECT_CLASS	The object for which properties were requested from Active Directory does not belong to the class requested.
0xC00E008D MQ_ERROR_MULTI_SORT_KEYS	The value of cCol in MQSORTSET cannot be greater than 1. Active Directory supports only a single sort key.
0xC00E008E MQ_ERROR_GC_NEEDED	An MSMQ Configuration (msmq) object with the GUID supplied cannot be created.
0xC00E008F MQ_ERROR_DS_BIND_ROOT_FOREST	Binding to the forest root failed. This error usually indicates a problem in the DNS configuration.
0xC00E0090 MQ_ERROR_DS_LOCAL_USER	A local user is authenticated as an anonymous user and cannot access Active Directory. The local user must log on as a domain user to access Active Directory.
0xC00E0091 MQ_ERROR_Q_ADS_PROPERTY_NOT_SUPPORTED	The column PROPID_Q_ADS_PATH is not supported for the API.
0xC00E0092 MQ_ERROR_BAD_XML_FORMAT	The given property is not a valid XML document.
0xC00E0093 MQ_ERROR_UNSUPPORTED_CLASS	The Active Directory object specified is not an instance of a supported class.
0xC00E0094 MQ_ERROR_UNINITIALIZED_OBJECT	The MSMQManagement object must be initialized before it is used.
0xC00E0095 MQ_ERROR_CANNOT_CREATE_PSC_OBJECTS	The create object operation cannot be performed because the object must be owned by a PSC.
0xC00E0096 MQ_ERROR_CANNOT_UPDATE_PSC_OBJECTS	The update operation cannot be performed because the designated object is owned by a PSC.
0xC00E0099 MQ_ERROR_RESOLVE_ADDRESS	Message Queuing is not able to resolve the address specified by the user. The address might be wrong, or the DNS lookup for the address failed.
0xC00E009A MQ_ERROR_TOO_MANY_PROPERTIES	Too many properties passed to the function. Message Queuing can process up to 128 properties in one call.

Return value/code	Description
0xC00E009B MQ_ERROR_MESSAGE_NOT_AUTHENTICATED	The queue only accepts authenticated messages.
0xC00E009C MQ_ERROR_MESSAGE_LOCKED_UNDER_TRANSACTION	The message is currently being processed under a transaction. Until the transaction outcome is determined, the message cannot be processed in any other transaction.
0xC00E0504 MQDS_UNKNOWN_SOURCE	The specified MSMQ Directory Service server in the directory change is unknown.

2.5 Message Properties for Digital Signatures

2.5.1 MSMQ 1.0 Digital Signature Properties

The MSMQ 1.0 digital signature MUST be calculated using the values of the following fields in the specified order:

- **MessagePropertiesHeader.CorrelationID**
- **MessagePropertiesHeader.ApplicationTag**
- **MessagePropertiesHeader.MessageBody**
- **MessagePropertiesHeader.Label**
- **UserHeader.ResponseQueue**
- **UserHeader.AdminQueue**

2.5.2 MSMQ 2.0 Digital Signature Properties

The MSMQ 2.0 digital signature MUST be calculated using the values of the following fields in the specified order:

- **MessagePropertiesHeader.CorrelationID**
- **MessagePropertiesHeader.ApplicationTag**
- **MessagePropertiesHeader.MessageBody**
- **MessagePropertiesHeader.Label**
- **UserHeader.ResponseQueue**
- **UserHeader.AdminQueue**
- **UserHeader.SourceQueueManager**
- (BYTE)**UserHeader.Flags.DM**
- (BYTE)**BaseHeader.Flags.PR**
- ((BYTE)**UserHeader.Flags.JP**) << 1 | ((BYTE)**UserHeader.Flags.JN**)

- ((BYTE)**MessagePropertiesHeader.Flags**) & 0x0F
- (USHORT)**MessagePropertiesHeader.MessageClass**
- (ULONG)**MessagePropertiesHeader.BodyType**
- **UserHeader.ConnectorType**
- **UserHeader.DestinationQueue**

2.5.3 MSMQ 3.0 Digital Signature Properties

The MSMQ 3.0 digital signature MUST be calculated using the values of the following fields in the specified order:

- **MessagePropertiesHeader.CorrelationID**
- **MessagePropertiesHeader.ApplicationTag**
- **MessagePropertiesHeader.MessageBody**
- **MessagePropertiesHeader.Label**
- **UserHeader.ResponseQueue**
- **UserHeader.AdminQueue**
- **MessagePropertiesHeader.ExtensionData**
- **MultiQueueFormatHeader.FormatNameList**
- **UserHeader.SourceQueueManager**
- (BYTE)**UserHeader.Flags.DM**
- (BYTE)**BaseHeader.Flags.PR**
- ((BYTE)**UserHeader.Flags.JP**) << 1 | ((BYTE)**UserHeader.Flags.JN**)
- ((BYTE)**MessagePropertiesHeader.Flags**) & 0x0F
- (USHORT)**MessagePropertiesHeader.MessageClass**
- (ULONG)**MessagePropertiesHeader.BodyType**
- **UserHeader.ConnectorType**

3 Structure Examples

None.

4 Security Considerations

None.

5 Appendix A: Full IDL

```
import "ms-dtyp.idl";

// forward declaration
typedef tag_inner_PROPVARIANT PROPVARIANT;

// basic type aliases
typedef unsigned long    PROPID;
typedef short            VARIANT_BOOL;


typedef struct {
  unsigned char rgb[16];
} XACTUOW;

typedef struct tagBLOB {
    unsigned long cbSize;
    [size_is(cbSize)]
    unsigned char *pBlobData;
} BLOB;

typedef struct tagCAUB
{
    unsigned long   cElems;
    [size_is( cElems )]
    unsigned char *  pElems;
} CAUB;

typedef struct tagCAUI
{
    unsigned long   cElems;
    [size_is( cElems )]
    unsigned short *  pElems;
} CAUI;

typedef struct tagCAL
{
    unsigned long   cElems;
    [size_is( cElems )]
    long  *  pElems;
} CAL;

typedef struct tagCAUL
{
    unsigned long   cElems;
    [size_is( cElems )]
    unsigned long *  pElems;
} CAUL;

typedef struct tagCAUH
{
    unsigned long   cElems;
    [size_is( cElems )]
    ULARGE_INTEGER *  pElems;
} CAUH;

typedef struct tagCACLSID
{
    unsigned long   cElems;
    [size_is( cElems )]
    GUID *  pElems;
} CACLSID;

typedef struct tagCALPWSTR
{
    unsigned long   cElems;
    [size_is( cElems )]
```

```
    [string] wchar_t **  pElems;
} CALPWSTR;

typedef struct tagCAPROPVARIANT
{
    unsigned long cElems;
    [size_is( cElems )]
    PROPVARIANT * pElems;
} CAPROPVARIANT;

typedef enum
{
    VT_EMPTY           = 0,
    VT_NULL            = 1,
    VT_I2              = 2,
    VT_I4              = 3,
    VT_BOOL            = 11,
    VT_VARIANT         = 12,
    VT_I1              = 16,
    VT_UI1             = 17,
    VT_UI2             = 18,
    VT_UI4             = 19,
    VT_I8              = 20,
    VT_UI8             = 21,
    VT_LPWSTR          = 31,
    VT_BLOB            = 65,
    VT_CLSID           = 72,

    VT_VECTOR          = 0x1000,
} VARENUM;

typedef unsigned short VARTYPE;

typedef struct _tag_inner_PROPVARIANT
{
    VARTYPE vt;
    UCHAR   wReserved1;
    UCHAR   wReserved2;
    ULONG   wReserved3;
    [switch_is(vt)] union
    {
        [case (VT_EMPTY, VT_NULL)];
        [case (VT_I1)]              CHAR            cVal;
        [case (VT_UI1)]             UCHAR           bVal;
        [case (VT_I2)]              SHORT           iVal;
        [case (VT_UI2)]             USHORT          uiVal;
        [case (VT_I4)]              LONG            lVal;
        [case (VT_UI4)]             ULONG           ulVal;
        [case (VT_I8)]              LARGE_INTEGER   hVal;
        [case (VT_UI8)]             ULARGE_INTEGER  uhVal;
        [case (VT_BOOL)]            VARIANT_BOOL    boolVal;
        [case (VT_CLSID)]           GUID *          puuid;
        [case (VT_BLOB)]            BLOB            blob;
        [case (VT_LPWSTR)]          [string] wchar_t * pwszVal;
        [case (VT_VECTOR|VT_UI1)]     CAUB          caub;
        [case (VT_VECTOR|VT_UI2)]     CAUI          caui;
        [case (VT_VECTOR|VT_I4)]      CAL           cal;
        [case (VT_VECTOR|VT_UI4)]     CAUL          caul;
        [case (VT_VECTOR|VT_UI8)]     CAUH          cauh;
        [case (VT_VECTOR|VT_CLSID)]   CACLSID       cauuid;
        [case (VT_VECTOR|VT_LPWSTR)]  CALPWSTR      calpwstr;
        [case (VT_VECTOR|VT_VARIANT)] CAPROPVARIANT capropvar;
    } _varUnion;
} tag_inner_PROPVARIANT;

typedef struct _DL_ID {
  GUID m_DlGuid;
  [string] wchar_t * m_pwzDomain;
} DL_ID;
```

```
typedef struct _MULTICAST_ID {
  ULONG m_address;
  ULONG m_port;
} MULTICAST_ID;

typedef struct _OBJECTID {
  GUID Lineage;
  DWORD Uniquifier;
} OBJECTID;

typedef  enum __QUEUE_FORMAT_TYPE
{
  QUEUE_FORMAT_TYPE_UNKNOWN = 0,
  QUEUE_FORMAT_TYPE_PUBLIC = 1,
  QUEUE_FORMAT_TYPE_PRIVATE = 2,
  QUEUE_FORMAT_TYPE_DIRECT = 3,
  QUEUE_FORMAT_TYPE_MACHINE = 4,
  QUEUE_FORMAT_TYPE_CONNECTOR = 5,
  QUEUE_FORMAT_TYPE_DL = 6,
  QUEUE_FORMAT_TYPE_MULTICAST = 7,
  QUEUE_FORMAT_TYPE_SUBQUEUE = 8
} QUEUE_FORMAT_TYPE;
typedef struct __QUEUE_FORMAT {
  unsigned char m_qft;
  unsigned char m_SuffixAndFlags;
  unsigned short m_reserved;
  [switch_is(m_qft)] union {
    [case(QUEUE_FORMAT_TYPE_UNKNOWN)]
      ; // No member is set. Selected when an m_qft value
        // of 0 is returned.
    [case(QUEUE_FORMAT_TYPE_PUBLIC)]
      GUID m_gPublicID;
    [case(QUEUE_FORMAT_TYPE_PRIVATE)]
      OBJECTID m_oPrivateID;
    [case(QUEUE_FORMAT_TYPE_DIRECT)]
      [string] wchar_t* m_pDirectID;
    [case(QUEUE_FORMAT_TYPE_MACHINE)]
      GUID m_gMachineID;
    [case(QUEUE_FORMAT_TYPE_CONNECTOR)]
      GUID m_GConnectorID;
    [case(QUEUE_FORMAT_TYPE_DL)]
      DL_ID m_DlID;
    [case(QUEUE_FORMAT_TYPE_MULTICAST)]
      MULTICAST_ID m_MulticastID;
    [case(QUEUE_FORMAT_TYPE_SUBQUEUE)]
      [string] wchar_t* m_pDirectSubqueueID;
  };
} QUEUE_FORMAT;
```

6 (Updated Section) Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

Windows Client

- Windows NT Workstation operating system
- Windows 2000 Professional operating system
- Windows XP operating system
- Windows Vista operating system
- Windows 7 operating system
- Windows 8 operating system
- Windows 8.1 operating system
- Windows 10 operating system

Windows Server

- Windows NT Server operating system
- Windows 2000 Server operating system
- Windows Server 2003 operating system
- Windows Server 2008 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 operating system
- Windows Server operating system
- Windows Server 2019 operating system
- Windows Server 2022 operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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 2.2.18.1.4: Padding bytes contain uninitialized values.

<2> Section 2.2.18.1.5.2: Padding bytes contain uninitialized values.

<3> Section 2.2.19.2: Negative source journaling is not supported by Windows NT operating system. Windows 7 and later client operating systems and Windows Server 2008 and later server operating systems do not perform source journaling for messages sent to administration queues, notification queues, and order queues.

<4> Section 2.2.19.2: Windows stores a copy of the message in the local dead-letter queue on failure to deliver. Transactional messages are copied to the local transactional dead-letter queue. The dead-letter queue is a system-generated queue and is implementation-dependent. Windows 7 and later client operating systems and Windows Server 2008 and later server operating systems do not perform source journaling for messages sent to administration queues, notification queues, and order queues.

<5> Section 2.2.19.2: Windows copies the message to the system journal queue. The journal queue is a system-generated queue and is implementation-dependent. Windows 7 and later client operating systems and Windows Server 2008 and later server operating systems do not perform source journaling for messages sent to administration queues, notification queues, and order queues.

<6> Section 2.2.19.3: 40-bit and 128-bit encryption is not supported by Windows NT. AES encryption is not supported by Windows NT, Windows 2000 operating system, Windows XP, or Windows Server 2003.

<7> Section 2.2.19.3: The SHA-1 hash algorithm is not supported by Windows NT and Windows 2000. The SHA-256 and SHA-512 hash algorithms are not supported by Windows NT, Windows 2000, Windows XP, and Windows Server 2003. As a security enhancement, the default hash algorithm has been set to a stronger one, as described in the following list:

- Windows NT and Windows 2000 default to MD5 (0x00008003).
- Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008 default to SHA-1 (0x00008004).
- Windows 7 and later client operating systems and Windows Server 2008 R2 and later server operating systems default to SHA-512 (0x0000800E).

<8> Section 2.2.19.3: Windows pads the MessagePropertiesHeader packet with uninitialized data.

<9> Section 2.2.20.1: The MultiQueueFormatHeader is not supported by Windows 2000 or Windows NT.

<10> Section 2.2.20.1: Padding bytes contain uninitialized values.

<11> Section 2.2.20.1: Padding bytes contain uninitialized values.

<12> Section 2.2.20.1: Padding bytes contain uninitialized values.

<13> Section 2.2.20.2~~<13> Section 2.2.20.2:~~: Padding bytes contain uninitialized values.

<14> Section 2.2.20.4: Coding errors present in Windows can prevent the **UserMsgSequenceNumber** from being set to the count of UserMessage Packets sent on a session.

<15> Section 2.2.20.4: If a destination queue manager encounters a memory allocation error while processing a message, it sets the window size to 1 in the outgoing SessionHeader header, which causes the destination queue manager to reduce its window to the same size. The source queue manager doubles the window size every 30 seconds until the window size returns to the default value of 64.

<16> Section 2.2.20.4: The maximum allowed value of this field can be configured by setting a value in the registry key HKEY_LOCAL_MACHINE\software\microsoft\msmq\parameters\MaxUnackedPacket. When this key is absent, the default maximum is 64.

<17> Section 2.2.20.4: The maximum allowed value of this field can be configured by setting a value in the registry key HKEY_LOCAL_MACHINE\software\microsoft\msmq\parameters\MaxUnackedPacket. When this key is absent, the default maximum is 64.

<18> Section 2.2.20.5: The **TransactionHeader.Flags.FM** and **TransactionHeader.Flags.LM** fields are not supported by Windows NT.

<19> Section 2.2.20.6: The default cryptographic provider for MSMQ on Windows XP and Windows Server 2003 is "Microsoft Base Cryptographic Provider v1.0" as described in [MSDN-BCP]. The default cryptographic provider for MSMQ on Windows Vista and later client operating systems and Windows Server 2008 and later server operating systems is "Microsoft AES Cryptographic Provider" as described in [MSDN-ACP].

<20> Section 2.2.20.6: Windows stores the queue manager public key in the MSMQ-Encrypt-Key attribute of the queue manager's MSMQ-Configuration object in Active Directory. During the queue manager startup, this value is queried once using the Lightweight Directory Access Protocol (LDAP).

<21> Section 2.2.20.6: The application can choose any of the currently available Microsoft cryptographic service providers (CSPs). The list of available Microsoft CSPs as well as the type and the name of each CSP are described in [MSDN-CSP].

<22> Section 2.2.20.6~~<22> Section 2.2.20.6:~~: The application can choose any of the currently available Microsoft cryptographic service providers (CSPs) listed in [MSDN-CSP].

<23> Section 2.2.20.8: To enable report messages, ensure that there is a registry key of type **DWORD** called HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSMQ\Parameters\Security\EnableReportMessages and that its value is set to 1.

<24> Section 2.3.1.5: When the property is not set, the storage size of a queue is limited only by the available disk space on the local computer or the computer quota. For Windows XP Professional operating system, there is no default computer quota. For Windows 2000, the default computer quota is 2 gigabytes. For the Windows Server 2003 family, the default computer quota is 8 gigabytes. For Windows Vista and later client operating systems and Windows Server 2008 and later server operating systems, the default computer quota is 4 gigabytes.

<25> Section 2.3.1.7: When the property is not set, the storage size of a queue is limited only by the available disk space on the local computer or by the journal quota. For Windows 2000, the default journal quota is 2 gigabytes. For Windows XP Professional, there is no default journal quota. For the Windows Server 2003 family, the default journal quota is 8 gigabytes. For Windows Vista and later client operating systems and Windows Server 2008 and later server operating systems, the default journal quota is 4 gigabytes.

<26> Section 2.3.1.16: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] and the collocated Message Queuing (MSMQ): Directory Service Protocol [MS-MQDS] server for directory service replication among MSMQ Directory Service servers.

<27> Section 2.3.1.25: The security descriptor ACE **AceType** fields are limited to those supported by Windows NT 4.0 operating system ([MS-DTYP] section 2.4.4.1).

<28> Section 2.3.2.11: This property is used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] and the collocated Message Queuing (MSMQ): Directory Service Protocol [MS-MQDS] server for directory service replication among MSMQ Directory Service servers.

<29> Section 2.3.2.15: Windows Vista and later client operating systems and Windows Server 2003 and later server operating systems accept an empty string.

<30> Section 2.3.2.15: Windows NT, Windows 95 operating system, Windows 98 operating system, and Windows 2000 require the following format in ABNF notation.

```
MachineType = WindowsLiteral Space OSType Space UInt "."
              UInt Space "(" BuildLiteral Space UInt "," Space Platform ")"
              Space "-" Space MSMQLiteral Space UInt "." UInt Space
              "(" BuildLiteral Space UInt ")"
WindowsLiteral = %x57 %x69 %x6e %x64 %x6f %x77 %x73
BuildLiteral = %x42 %x75 %x69 %x6c %x64
MSMQLiteral = %x4d %x53 %x4d %x51
OSType = *(%x20-7E)
Platform = *(%x20-7E)
UInt = *(%x30-39)
Space = %x20
```

<31> Section 2.3.2.36: The security descriptor ACE **AceType** fields are limited to those supported by Windows NT 4.0 ([MS-DTYP] section 2.4.4.1).

<32> Section 2.3.3.7: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] for directory service replication among MSMQ Directory Service servers.

<33> Section 2.3.3.13: The security descriptor ACE **AceType** fields are limited to those supported by Windows NT 4.0 ([MS-DTYP] section 2.4.4.1).

<34> Section 2.3.4.4: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] for directory service replication among MSMQ Directory Service servers.

<35> Section 2.3.4.6: The security descriptor ACE **AceType** fields are limited to those supported by Windows NT 4.0 ([MS-DTYP] section 2.4.4.1).

<36> Section 2.3.5.2: Windows 2000 Professional and later client operating systems and Windows 2000 Server and later server operating systems use an internal default equivalent to setting this value to 0x00.

<37> Section 2.3.5.7: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] for directory service replication among MSMQ Directory Service servers.

<38> Section 2.3.5.17: The security descriptor ACE **AceType** fields are limited to those supported by Windows NT 4.0 ([MS-DTYP] section 2.4.4.1).

<39> Section 2.3.6.2: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] for directory service replication among MSMQ Directory Service servers.

<40> Section 2.3.7.4: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode archived in the Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] for directory service replication among MSMQ Directory Service servers.

<41> Section 2.3.8.6: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] for directory service replication among MSMQ Directory Service servers.

<42> Section 2.3.9.5: The security descriptor ACE **AceType** fields are limited to those supported by Windows NT 4.0 ([MS-DTYP] section 2.4.4.1).

<43> Section 2.3.13.2: This property was used only by Windows NT and Windows 2000 operating in MSMQ mixed-mode in the archived Message Queuing (MSMQ): Directory Service Replication Protocol [MC-MQDSRP] for directory service replication among MSMQ Directory Service servers.

<44> Section 2.4: For more information about MSMQ-specific **HRESULT** values, see [MSDN-MQEIC].

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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.
- A document revision that captures changes to protocol functionality.

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 **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
6 Appendix B: Product Behavior	Updated for this version of Windows Server.	Major

8 Index

L

M

N

O

P

PROPID_ MQU_ID 97
PROPID_CN_GUID 87
PROPID_CN_NAME 87
PROPID_CN_PARTITIONID 87
PROPID_CN_PROTOCOLID 86
PROPID_CN_SEQNUM 87
PROPID_COM_ACCOUNT_CONTROL 98
PROPID_COM_DIGEST 99
PROPID_COM_DNS_HOSTNAME 98
PROPID_COM_FULL_PATH 98
PROPID_COM_ID 99
PROPID_COM_SAM_ACCOUNT 98
PROPID_COM_SID 98
PROPID_COM_SIGN_CERT 98
PROPID_D_IDENTIFIER 109
PROPID_D_PARTITIONID 108
PROPID_D_SEQNUM 108
PROPID_E_ ID 89
PROPID_E_CIPHER_MODE 90
PROPID_E_CRL 89
PROPID_E_CSP_NAME 88
PROPID_E_CSP_TYPE 89
PROPID_E_ENCRYPT_ALG 89
PROPID_E_HASH_ALG 90
PROPID_E_LONG_LIVE 90
PROPID_E_NAME 88
PROPID_E_NAMESTYLE 88
PROPID_E_PARTITIONID 89
PROPID_E_PECNAME 88
PROPID_E_S_INTERVAL1 88
PROPID_E_S_INTERVAL2 89
PROPID_E_SECURITY 90
PROPID_E_SEQNUM 89
PROPID_E_SIGN_ALG 90
PROPID_E_VERSION 90
PROPID_L_ACTUAL_COST 93
PROPID_L_COST 92
PROPID_L_DESCRIPTION 93
PROPID_L_FULL_PATH 93
PROPID_L_GATES 94
PROPID_L_GATES_DN 92
PROPID_L_ID 92
PROPID_L_NEIGHBOR1 92
PROPID_L_NEIGHBOR1_DN 93
PROPID_L_NEIGHBOR2 92
PROPID_L_NEIGHBOR2_DN 93
PROPID_L_PARTITIONID 92
PROPID_L_SEQNUM 92
PROPID_MGMT_MSMQ_ACTIVEQUEUES 99
PROPID_MGMT_MSMQ_BYTES_IN_ALL_QUEUES 100
PROPID_MGMT_MSMQ_CONNECTED 100
PROPID_MGMT_MSMQ_DSSERVER 99
PROPID_MGMT_MSMQ_PRIVATEQ 99
PROPID_MGMT_MSMQ_TYPE 100
PROPID_MGMT_QUEUE_ SUBQUEUE_COUNT 108
PROPID_MGMT_QUEUE_ SUBQUEUE_NAMES 108
PROPID_MGMT_QUEUE_BYTES_IN_JOURNAL 103
PROPID_MGMT_QUEUE_BYTES_IN_QUEUE 102
PROPID_MGMT_QUEUE_CONNECTION_HISTORY 106
PROPID_MGMT_QUEUE_EOD_FIRST_NON_ACK 104
PROPID_MGMT_QUEUE_EOD_LAST_ACK 104
PROPID_MGMT_QUEUE_EOD_LAST_ACK_COUNT 104
PROPID_MGMT_QUEUE_EOD_LAST_ACK_TIME 104
PROPID_MGMT_QUEUE_EOD_LAST_NON_ACK 104
PROPID_MGMT_QUEUE_EOD_NEXT_SEQ 104

Q

R

S

T

U

V

X