

[MC-MQAC]: Message Queuing (MSMQ): ActiveX Client Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- Microsoft does not claim any trade secret rights in this documentation.
- Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. If you are interested in obtaining a patent license, please contact protocol@microsoft.com.
- The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
08/10/2007	0.1	Major	Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	1.0	Major	Updated and revised the technical content.
11/30/2007	2.0	Major	Performed XML markup and page conversions;

Date	Revision History	Revision Class	Comments
			restructured content.
01/25/2008	2.0.1	Editorial	Revised and edited the technical content.

Table of Contents

1	Introduction	13
1.1	Glossary	13
1.2	References	15
1.2.1	Normative References	15
1.2.2	Informative References.....	17
1.3	Protocol Overview (Synopsis).....	17
1.3.1	Scenario: Retrieving the count of [Messages] in a [Queue]	20
1.3.2	Scenario: Retrieving [EODTargetInfos] for an [Application Queue]	20
1.3.3	Scenario: Pausing an [Outgoing Queue]	21
1.3.4	Scenario: Discovering [Directory Queue]s in the [Directory].....	22
1.3.5	Scenario: Receiving a message from a queue asynchronously via event callbacks	23
1.3.6	Scenario: Sending a multicast message	24
1.3.7	Scenario: Sending a message with an internal transaction	25
1.3.8	Scenario: Sending a message with an external transaction	26
1.3.9	Scenario: Sending [Message] to a [Queue].....	28
1.4	Relationship to Other Protocols.....	28
1.4.1	Message Transfer.....	29
1.4.2	Directory Access	29
1.4.3	Distributed Transaction Coordination	29
1.5	Prerequisites/Preconditions	29
1.6	Applicability Statement	30
1.7	Versioning and Capability Negotiation.....	30
1.8	Vendor-Extensible Fields	30
1.9	Standards Assignments.....	31
2	Messages	35
2.1	Transport	35
2.2	Common Data Types	35
2.2.1	OLE Automation Data Types	35
2.2.2	Enumerations.....	35
2.2.2.1	MQTRANSACTION.....	36
2.2.2.2	MQSHARE	37
2.2.2.3	MQACCESS.....	37
2.2.2.4	MQJOURNAL	38
2.2.2.5	MQTRANSACTIONAL	39
2.2.2.6	MQAUTHENTICATE	39
2.2.2.7	MQPRIVLEVEL.....	40
2.2.2.8	MQMSGCURSOR.....	40
2.2.2.9	MQMSGCLASS	41
2.2.2.10	MQMSGDELIVERY	43
2.2.2.11	MQMSGACKNOWLEDGEMENT	44
2.2.2.12	MQMSGJOURNAL.....	45
2.2.2.13	MQMSGTRACE	46
2.2.2.14	MQMSGSENDERIDTYPE	46
2.2.2.15	MQMSGPRIVLEVEL.....	47
2.2.2.16	MQMSGAUTHLEVEL	47
2.2.2.17	MQMSGAUTHENTICATION	48
2.2.2.18	MQCALG	49
2.2.2.19	QUEUE_STATE	50
2.2.2.20	RELOPS	51
2.2.2.21	XACTTC	52
2.2.3	Data Collections	53

2.2.3.1	EODTargetInfo	53
2.2.3.2	EODSourceInfo	53
2.2.3.3	SequenceInfoCollection	55
3	Protocol Details	56
3.1	Common Implementation Details.....	56
3.1.1	Abstract Data Model	56
3.1.1.1	Directory.....	56
3.1.1.2	Directory Users Table.....	57
3.1.1.3	Directory User	57
3.1.1.4	Certificates Table	57
3.1.1.5	Certificate	57
3.1.1.6	Internal Certificate	57
3.1.1.7	External Certificate	57
3.1.1.8	Directory Queue Managers Table	58
3.1.1.9	Directory Queue Manager	58
3.1.1.10	Directory Queues Table	58
3.1.1.11	Directory Queue.....	58
3.1.1.12	Queue Manager.....	59
3.1.1.13	Local Queue Manager.....	60
3.1.1.14	Queues Table.....	60
3.1.1.15	Queue	60
3.1.1.16	Application Queue	60
3.1.1.17	Public Queue	62
3.1.1.18	Private Queue.....	62
3.1.1.19	Outgoing Queue	62
3.1.1.20	System Queue	62
3.1.1.21	Journal Queue	63
3.1.1.22	Messages Table.....	63
3.1.1.23	Message	63
3.1.1.24	Open Queues Table	67
3.1.1.25	Open Queue	67
3.1.1.26	Enlisted Transactions Table.....	67
3.1.1.27	Enlisted Transaction.....	67
3.1.1.28	Transactional Operation	67
3.1.2	Timers	68
3.1.2.1	Time To Reach Queue	68
3.1.2.2	Time To Be Received	68
3.1.3	Initialization.....	69
3.1.4	Message Processing Events and Sequencing Rules	69
3.1.4.1	Security	69
3.1.4.2	Optional Arguments.....	69
3.1.4.3	Out-Parameters and Errors	69
3.1.5	Timer Events.....	69
3.1.5.1	Time To Reach Queue Expiration	69
3.1.5.2	Time To Be Received Expiration.....	70
3.1.6	Other Local Events.....	70
3.1.6.1	Transaction Commit.....	70
3.1.6.2	Transaction Abort.....	71
3.2	MSMQApplication Coclasm	71
3.2.1	Abstract Data Model	72
3.2.2	Timers	72
3.2.3	Initialization.....	72
3.2.4	Message Processing Events and Sequencing Rules	72
3.2.4.1	IMSMQApplication Interface	73

3.2.4.1.1	MachineIdOfMachineName (Opnum 7)	74
3.2.4.2	IMSMQApplication2 Interface	75
3.2.4.2.1	RegisterCertificate (Opnum 8)	75
3.2.4.2.2	MachineNameOfMachineId (Opnum 9)	77
3.2.4.2.3	MSMQVersionMajor (Opnum 10)	78
3.2.4.2.4	MSMQVersionMinor (Opnum 11)	78
3.2.4.2.5	MSMQVersionBuild (Opnum 12)	79
3.2.4.2.6	IsDsEnabled (Opnum 13)	79
3.2.4.2.7	Properties (Opnum 14)	80
3.2.4.3	IMSMQApplication3 Interface	80
3.2.4.3.1	ActiveQueues (Opnum 15)	81
3.2.4.3.2	PrivateQueues (Opnum 16)	82
3.2.4.3.3	DirectoryServiceServer (Opnum 17)	83
3.2.4.3.4	IsConnected (Opnum 18)	83
3.2.4.3.5	BytesInAllQueues (Opnum 19)	84
3.2.4.3.6	Machine (Opnum 20)	84
3.2.4.3.7	Machine (Opnum 21)	85
3.2.4.3.8	Connect (Opnum 22)	85
3.2.4.3.9	Disconnect (Opnum 23)	86
3.2.4.3.10	Tidy (Opnum 24)	87
3.2.5	Timer Events	87
3.2.6	Other Local Events	87
3.3	MSMQManagement Coclass	87
3.3.1	Abstract Data Model	87
3.3.2	Timers	88
3.3.3	Initialization	88
3.3.4	Message Processing Events and Sequencing Rules	88
3.3.4.1	IMSMQManagement Interface	88
3.3.4.1.1	Init (Opnum 7)	89
3.3.4.1.2	FormatName (Opnum 8)	91
3.3.4.1.3	Machine (Opnum 9)	91
3.3.4.1.4	MessageCount (Opnum 10)	92
3.3.4.1.5	ForeignStatus (Opnum 11)	93
3.3.4.1.6	QueueType (Opnum 12)	94
3.3.4.1.7	IsLocal (Opnum 13)	95
3.3.4.1.8	TransactionalStatus (Opnum 14)	96
3.3.4.1.9	BytesInQueue (Opnum 15)	98
3.3.5	Timer Events	98
3.3.6	Other Local Events	99
3.4	MSMQQueueManagement Coclass	99
3.4.1	Abstract Data Model	99
3.4.2	Timers	99
3.4.3	Initialization	99
3.4.4	Message Processing Events and Sequencing Rules	99
3.4.4.1	IMSMQQueueManagement Interface	99
3.4.4.1.1	JournalMessageCount (Opnum 16)	100
3.4.4.1.2	BytesInJournal (Opnum 17)	101
3.4.4.1.3	EodGetReceiveInfo (Opnum 18)	102
3.4.5	Timer Events	103
3.4.6	Other Local Events	103
3.5	MSMQOutgoingQueueManagement Coclass	103
3.5.1	Abstract Data Model	103
3.5.2	Timers	103
3.5.3	Initialization	104
3.5.4	Message Processing Events and Sequencing Rules	104

3.5.4.1	IMSMQOutgoingQueueManagement Interface	104
3.5.4.1.1	State (Opnum 16).....	105
3.5.4.1.2	NextHops (Opnum 17).....	106
3.5.4.1.3	EodGetSendInfo (Opnum 18).....	107
3.5.4.1.4	Resume (Opnum 19).....	107
3.5.4.1.5	Pause (Opnum 20).....	108
3.5.4.1.6	EodResend (Opnum 21).....	109
3.5.5	Timer Events.....	110
3.5.6	Other Local Events.....	110
3.6	MSMQTransactionDispenser Coclass	110
3.6.1	Abstract Data Model	110
3.6.2	Timers	110
3.6.3	Initialization	111
3.6.4	Message Processing Events and Sequencing Rules	111
3.6.4.1	IMSMQTransactionDispenser3 Interface	111
3.6.4.1.1	BeginTransaction (Opnum 7)	112
3.6.4.1.2	Properties (Opnum 8).....	112
3.6.5	Timer Events.....	112
3.6.6	Other Local Events.....	113
3.7	MSMQCoordinatedTransactionDispenser Coclass	113
3.7.1	Abstract Data Model	113
3.7.2	Timers	113
3.7.3	Initialization	113
3.7.4	Message Processing Events and Sequencing Rules	113
3.7.4.1	IMSMQCoordinatedTransactionDispenser3 Interface	113
3.7.4.1.1	BeginTransaction (Opnum 7)	114
3.7.4.1.2	Properties (Opnum 8).....	115
3.7.5	Timer Events.....	115
3.7.6	Other Local Events.....	115
3.8	ITransaction Implementation Class	115
3.8.1	Abstract Data Model	115
3.8.2	Timers	115
3.8.3	Initialization	115
3.8.4	Message Processing Events and Sequencing Rules	116
3.8.4.1	ITransaction Interface.....	116
3.8.4.1.1	Commit (Opnum 3).....	116
3.8.4.1.2	Abort (Opnum 4).....	117
3.8.4.1.3	GetTransactionInfo (Opnum 5)	118
3.8.5	Timer Events.....	119
3.8.6	Other Local Events.....	120
3.9	MSMQTransaction Coclass	120
3.9.1	Abstract Data Model	120
3.9.2	Timers	120
3.9.3	Initialization	120
3.9.4	Message Processing Events and Sequencing Rules	121
3.9.4.1	IMSMQTransaction Interface	121
3.9.4.1.1	Transaction (Opnum 7).....	122
3.9.4.1.2	Commit (Opnum 8).....	122
3.9.4.1.3	Abort (Opnum 9).....	123
3.9.4.2	IMSMQTransaction2 Interface	123
3.9.4.2.1	InitNew (Opnum 10)	124
3.9.4.2.2	Properties (Opnum 11)	124
3.9.4.3	IMSMQTransaction3 Interface	125
3.9.4.3.1	ITransaction (Opnum 12).....	125
3.9.5	Timer Events.....	125

3.9.6	Other Local Events	125
3.10	MSMQQueueInfo Coclass	125
3.10.1	Abstract Data Model	126
3.10.2	Timers	128
3.10.3	Initialization	128
3.10.4	Message Processing Events and Sequencing Rules	129
3.10.4.1	IMSMQQueueInfo4 Interface	131
3.10.4.1.1	QueueGuid (Opnum 7)	134
3.10.4.1.2	ServiceTypeGuid (Opnum 8)	135
3.10.4.1.3	ServiceTypeGuid (Opnum 9)	135
3.10.4.1.4	Label (Opnum 10)	135
3.10.4.1.5	Label (Opnum 11)	136
3.10.4.1.6	PathName (Opnum 12)	136
3.10.4.1.7	PathName (Opnum 13)	137
3.10.4.1.8	FormatName (Opnum 14)	137
3.10.4.1.9	FormatName (Opnum 15)	137
3.10.4.1.10	IsTransactional (Opnum 16)	138
3.10.4.1.11	PrivLevel (Opnum 17)	138
3.10.4.1.12	PrivLevel (Opnum 18)	139
3.10.4.1.13	Journal (Opnum 19)	139
3.10.4.1.14	Journal (Opnum 20)	140
3.10.4.1.15	Quota (Opnum 21)	140
3.10.4.1.16	Quota (Opnum 22)	140
3.10.4.1.17	BasePriority (Opnum 23)	141
3.10.4.1.18	BasePriority (Opnum 24)	141
3.10.4.1.19	CreateTime (Opnum 25)	142
3.10.4.1.20	ModifyTime (Opnum 26)	142
3.10.4.1.21	Authenticate (Opnum 27)	143
3.10.4.1.22	Authenticate (Opnum 28)	143
3.10.4.1.23	JournalQuota (Opnum 29)	144
3.10.4.1.24	JournalQuota (Opnum 30)	144
3.10.4.1.25	IsWorldReadable (Opnum 31)	144
3.10.4.1.26	Create (Opnum 32)	145
3.10.4.1.27	Delete (Opnum 33)	147
3.10.4.1.28	Open (Opnum 34)	148
3.10.4.1.29	Refresh (Opnum 35)	150
3.10.4.1.30	Update (Opnum 36)	152
3.10.4.1.31	PathNameDNS (Opnum 37)	153
3.10.4.1.32	Properties (Opnum 38)	154
3.10.4.1.33	Security (Opnum 39)	154
3.10.4.1.34	Security (Opnum 40)	154
3.10.4.1.35	IsTransactional2 (Opnum 41)	155
3.10.4.1.36	IsWorldReadable (Opnum 42)	155
3.10.4.1.37	MulticastAddress (Opnum 43)	155
3.10.4.1.38	MulticastAddress (Opnum 44)	156
3.10.4.1.39	ADsPath (Opnum 45)	156
3.10.5	Timer Events	157
3.10.6	Other Local Events	157
3.11	MSMQQueue Coclass	157
3.11.1	Abstract Data Model	158
3.11.1.1	Object State Machine	158
3.11.1.2	Cursor State Machine	159
3.11.2	Timers	159
3.11.3	Initialization	159
3.11.4	Message Processing Events and Sequencing Rules	160

3.11.4.1	IMSMQQueue4 Interface	162
3.11.4.1.1	Access (Opnum 7)	165
3.11.4.1.2	ShareMode (Opnum 8)	165
3.11.4.1.3	QueueInfo (Opnum 9)	166
3.11.4.1.4	Handle (Opnum 10)	166
3.11.4.1.5	IsOpen (Opnum 11)	167
3.11.4.1.6	Close (Opnum 12).....	167
3.11.4.1.7	Receive_v1 (Opnum 13)	168
3.11.4.1.8	Peek_v1 (Opnum 14)	171
3.11.4.1.9	EnableNotification (Opnum 15)	173
3.11.4.1.10	Reset (Opnum 16).....	176
3.11.4.1.11	ReceiveCurrent_v1 (Opnum 17)	176
3.11.4.1.12	PeekNext_v1 (Opnum 18)	179
3.11.4.1.13	PeekCurrent_v1 (Opnum 19)	181
3.11.4.1.14	Receive (Opnum 20).....	183
3.11.4.1.15	Peek (Opnum 21).....	187
3.11.4.1.16	ReceiveCurrent (Opnum 22)	189
3.11.4.1.17	PeekNext (Opnum 23)	193
3.11.4.1.18	PeekCurrent (Opnum 24)	195
3.11.4.1.19	Properties (Opnum 25)	197
3.11.4.1.20	Handle2 (Opnum 26)	198
3.11.4.1.21	ReceiveByLookupId (Opnum 27)	198
3.11.4.1.22	ReceiveNextByLookupId (Opnum 28)	201
3.11.4.1.23	ReceivePreviousByLookupId (Opnum 29).....	204
3.11.4.1.24	ReceiveFirstByLookupId (Opnum 30)	207
3.11.4.1.25	ReceiveLastByLookupId (Opnum 31).....	210
3.11.4.1.26	PeekByLookupId (Opnum 32)	213
3.11.4.1.27	PeekNextByLookupId (Opnum 33)	215
3.11.4.1.28	PeekPreviousByLookupId (Opnum 34).....	217
3.11.4.1.29	PeekFirstByLookupId (Opnum 35)	219
3.11.4.1.30	PeekLastByLookupId (Opnum 36).....	221
3.11.4.1.31	Purge (Opnum 37)	222
3.11.4.1.32	IsOpen2 (Opnum 38).....	223
3.11.4.1.33	ReceiveByLookupIdAllowPeek (Opnum 39).....	224
3.11.5	Timer Events.....	227
3.11.6	Other Local Events.....	227
3.12	MSMQDestination Coclass.....	227
3.12.1	Abstract Data Model	227
3.12.2	Timers	228
3.12.3	Initialization.....	228
3.12.4	Message Processing Events and Sequencing Rules	228
3.12.4.1	IMSMQDestination Interface.....	228
3.12.4.1.1	Open (Opnum 7)	229
3.12.4.1.2	Close (Opnum 8)	231
3.12.4.1.3	IsOpen (Opnum 9).....	231
3.12.4.1.4	IADs (Opnum 10)	232
3.12.4.1.5	IADs (Opnum 11)	232
3.12.4.1.6	ADsPath (Opnum 12)	232
3.12.4.1.7	ADsPath (Opnum 13)	233
3.12.4.1.8	PathName (Opnum 14).....	233
3.12.4.1.9	PathName (Opnum 15).....	234
3.12.4.1.10	FormatName (Opnum 16)	234
3.12.4.1.11	FormatName (Opnum 17)	235
3.12.4.1.12	Destinations (Opnum 18)	235
3.12.4.1.13	Destinations (Opnum 19)	236

3.12.4.1.14 Properties (Opnum 20)	236
3.12.4.2 IMSMQPrivateDestination Interface	236
3.12.4.2.1 Handle (Opnum 7)	237
3.12.4.2.2 Handle (Opnum 8)	237
3.12.5 Timer Events.....	238
3.12.6 Other Local Events	238
3.13 MSMQQuery Coclass	238
3.13.1 Abstract Data Model	238
3.13.2 Timers	238
3.13.3 Initialization.....	238
3.13.4 Message Processing Events and Sequencing Rules	238
3.13.4.1 IMSMQQuery4 Interface	239
3.13.4.1.1 LookupQueue_v2 (Opnum 7).....	239
3.13.4.1.2 Properties (Opnum 8).....	241
3.13.4.1.3 LookupQueue (Opnum 9).....	242
3.13.5 Timer Events.....	244
3.13.6 Other Local Events	244
3.14 MSMQQueueInfos Coclass.....	244
3.14.1 Abstract Data Model	244
3.14.2 Timers	244
3.14.3 Initialization.....	245
3.14.4 Message Processing Events and Sequencing Rules	245
3.14.4.1 IMSMQQueueInfos4 Interface.....	245
3.14.4.1.1 Reset (Opnum 7)	246
3.14.4.1.2 Next (Opnum 8)	246
3.14.4.1.3 Properties (Opnum 9).....	247
3.14.5 Timer Events.....	247
3.14.6 Other Local Events	248
3.15 MSMQCollection Coclass	248
3.15.1 Abstract Data Model	248
3.15.2 Timers	248
3.15.3 Initialization.....	248
3.15.4 Message Processing Events and Sequencing Rules	248
3.15.4.1 IMSMQCollection Interface.....	248
3.15.4.1.1 Item (Opnum 7)	249
3.15.4.1.2 Count (Opnum 8).....	249
3.15.4.1.3 _NewEnum (Opnum 9)	250
3.15.5 Timer Events.....	250
3.15.6 Other Local Events	250
3.16 MSMQEvent Coclass.....	250
3.16.1 Abstract Data Model	251
3.16.2 Timers	251
3.16.3 Initialization.....	251
3.16.4 Message Processing Events and Sequencing Rules	251
3.16.4.1 IMSMQEvent3 Interface.....	251
3.16.4.1.1 Properties (Opnum 7).....	252
3.16.4.2 IMSMQPrivateEvent Interface	252
3.16.4.2.1 Hwnd (Opnum 7)	253
3.16.4.2.2 FireArrivedEvent (Opnum 8)	253
3.16.4.2.3 FireArrivedErrorEvent (Opnum 9)	254
3.16.4.3 _DMSMQEventEvents Interface	254
3.16.4.3.1 Arrived (Opnum 7).....	254
3.16.4.3.2 ArrivedError (Opnum 8).....	255
3.16.4.4 IConnectionPoint Interface.....	255
3.16.4.4.1 GetConnectionInterface (Opnum 3)	256

3.16.4.4.2	GetConnectionPointContainer (Opnum 4)	256
3.16.4.4.3	Advise (Opnum 5)	257
3.16.4.4.4	Unadvise (Opnum 6)	258
3.16.4.4.5	EnumConnections (Opnum 7)	258
3.16.4.5	IConnectionPointContainer Interface	259
3.16.4.5.1	EnumConnectionPoints (Opnum 3)	259
3.16.4.5.2	FindConnectionPoint (Opnum 4)	260
3.16.5	Timer Events	261
3.16.6	Other Local Events	261
3.17	MSMQMessage Coclass	261
3.17.1	Abstract Data Model	261
3.17.2	Timers	262
3.17.3	Initialization	262
3.17.4	Message Processing Events and Sequencing Rules	265
3.17.4.1	IMSMQMessage4 Interface	269
3.17.4.1.1	Class (Opnum 7)	276
3.17.4.1.2	PrivLevel (Opnum 8)	277
3.17.4.1.3	PrivLevel (Opnum 9)	277
3.17.4.1.4	AuthLevel (Opnum 10)	277
3.17.4.1.5	AuthLevel (Opnum 11)	278
3.17.4.1.6	IsAuthenticated (Opnum 12)	278
3.17.4.1.7	Delivery (Opnum 13)	279
3.17.4.1.8	Delivery (Opnum 14)	279
3.17.4.1.9	Trace (Opnum 15)	280
3.17.4.1.10	Trace (Opnum 16)	280
3.17.4.1.11	Priority (Opnum 17)	280
3.17.4.1.12	Priority (Opnum 18)	281
3.17.4.1.13	Journal (Opnum 19)	281
3.17.4.1.14	Journal (Opnum 20)	282
3.17.4.1.15	ResponseQueueInfo_v1 (Opnum 21)	282
3.17.4.1.16	ResponseQueueInfo_v1 (Opnum 22)	283
3.17.4.1.17	AppSpecific (Opnum 23)	283
3.17.4.1.18	AppSpecific (Opnum 24)	283
3.17.4.1.19	SourceMachineGuid (Opnum 25)	284
3.17.4.1.20	BodyLength (Opnum 26)	284
3.17.4.1.21	Body (Opnum 27)	285
3.17.4.1.22	Body (Opnum 28)	285
3.17.4.1.23	AdminQueueInfo_v1 (Opnum 29)	286
3.17.4.1.24	AdminQueueInfo (Opnum 30)	286
3.17.4.1.25	Id (Opnum 31)	287
3.17.4.1.26	CorrelationId (Opnum 32)	287
3.17.4.1.27	CorrelationId (Opnum 33)	288
3.17.4.1.28	Ack (Opnum 34)	288
3.17.4.1.29	Ack (Opnum 35)	288
3.17.4.1.30	Label (Opnum 36)	289
3.17.4.1.31	Label (Opnum 37)	289
3.17.4.1.32	MaxTimeToReachQueue (Opnum 38)	290
3.17.4.1.33	MaxTimeToReachQueue (Opnum 39)	290
3.17.4.1.34	MaxTimeToReceive (Opnum 40)	291
3.17.4.1.35	MaxTimeToReceive (Opnum 41)	291
3.17.4.1.36	HashAlgorithm (Opnum 42)	291
3.17.4.1.37	HashAlgorithm (Opnum 43)	292
3.17.4.1.38	EncryptAlgorithm (Opnum 44)	292
3.17.4.1.39	EncryptAlgorithm (Opnum 45)	293
3.17.4.1.40	SentTime (Opnum 46)	293

3.17.4.1.41	ArrivedTime (Opnum 47).....	293
3.17.4.1.42	DestinationQueueInfo (Opnum 48)	294
3.17.4.1.43	SenderCertificate (Opnum 49)	294
3.17.4.1.44	SenderCertificate (Opnum 50)	295
3.17.4.1.45	SenderId (Opnum 51).....	295
3.17.4.1.46	SenderIdType (Opnum 52)	295
3.17.4.1.47	SenderIdType (Opnum 53)	296
3.17.4.1.48	Send (Opnum 54)	296
3.17.4.1.49	AttachCurrentSecurityContext (Opnum 55)	300
3.17.4.1.50	SenderVersion (Opnum 56)	301
3.17.4.1.51	Extension (Opnum 57)	301
3.17.4.1.52	Extension (Opnum 58)	302
3.17.4.1.53	ConnectorTypeGuid (Opnum 59)	302
3.17.4.1.54	ConnectorTypeGuid (Opnum 60)	303
3.17.4.1.55	TransactionStatusQueueInfo (Opnum 61)	303
3.17.4.1.56	DestinationSymmetricKey (Opnum 62).....	304
3.17.4.1.57	DestinationSymmetricKey (Opnum 63).....	304
3.17.4.1.58	Signature (Opnum 64)	305
3.17.4.1.59	Signature (Opnum 65)	305
3.17.4.1.60	AuthenticationProviderType (Opnum 66)	306
3.17.4.1.61	AuthenticationProviderType (Opnum 67)	306
3.17.4.1.62	AuthenticationProviderName (Opnum 68)	306
3.17.4.1.63	AuthenticationProviderName (Opnum 69)	307
3.17.4.1.64	SenderId (Opnum 70).....	307
3.17.4.1.65	MsgClass (Opnum 71).....	308
3.17.4.1.66	MsgClass (Opnum 72).....	308
3.17.4.1.67	Properties (Opnum 73)	309
3.17.4.1.68	TransactionId (Opnum 74)	309
3.17.4.1.69	IsFirstInTransaction (Opnum 75)	309
3.17.4.1.70	IsLastInTransaction (Opnum 76)	310
3.17.4.1.71	ResponseQueueInfo_v2 (Opnum 77).....	310
3.17.4.1.72	ResponseQueueInfo_v2 (Opnum 78).....	311
3.17.4.1.73	AdminQueueInfo_v2 (Opnum 79)	311
3.17.4.1.74	AdminQueueInfo_v2 (Opnum 80)	312
3.17.4.1.75	ReceivedAuthenticationLevel (Opnum 81)	312
3.17.4.1.76	ResponseQueueInfo (Opnum 82).....	312
3.17.4.1.77	ResponseQueueInfo (Opnum 83).....	313
3.17.4.1.78	AdminQueueInfo (Opnum 84)	313
3.17.4.1.79	AdminQueueInfo (Opnum 85)	314
3.17.4.1.80	ResponseDestination (Opnum 86).....	314
3.17.4.1.81	ResponseDestination (Opnum 87).....	315
3.17.4.1.82	Destination (Opnum 88).....	315
3.17.4.1.83	LookupId (Opnum 89)	316
3.17.4.1.84	IsAuthenticated2 (Opnum 90).....	316
3.17.4.1.85	IsFirstInTransaction2 (Opnum 91)	317
3.17.4.1.86	IsLastInTransaction2 (Opnum 92)	317
3.17.4.1.87	AttachCurrentSecurityContext2 (Opnum 93)	317
3.17.4.1.88	SoapEnvelope (Opnum 94).....	318
3.17.4.1.89	CompoundMessage (Opnum 95).....	319
3.17.4.1.90	SoapHeader (Opnum 96)	319
3.17.4.1.91	SoapBody (Opnum 97)	319
3.17.5	Timer Events.....	320
3.17.6	Other Local Events.....	320

4 Protocol Examples321

5	Security	322
5.1	Security Considerations for Implementers	322
5.2	Index of Security Parameters	322
6	Appendix A: Full IDL	323
7	Appendix B: Windows Behavior	360
8	Index.....	373

1 Introduction

This document specifies the Message Queuing (MSMQ): ActiveX Client Protocol, a collection of Distributed Component Object Model (DCOM) [\[MS-DCOM\]](#) interfaces that exposes **message** queuing functionality for use by client applications. Operations that a client would perform using this protocol include:

- Queuing system management
- Queue management
- Queue discovery
- **Transaction** management
- Sending and receiving of messages

Notational Conventions

The following notational conventions are used throughout this document:

System abstract data model elements, as defined in section [3.1](#), appear in square brackets [].

The dot "." notation is used to refer to a property of a system abstract data model element. If [A] refers to an element of the system abstract data model, [A].[Property] denotes the [Property] property of the element [A].

Elements of the abstract data models that are defined in section [3](#) are referred to as "instance variables." The names of instance variables are formatted as *italic*.

Monospaced typeface is used for all method signatures and **IDL** declarations.

1.1 Glossary

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

Administrator
Anonymous User
Augmented Backus-Naur Form (ABNF)
Certificate
Certificate Store
CLSID
Commit Request
Computer
Computer Name
Coordinated Universal Time (UTC)
Directory Service
DNS Name
Globally Unique Identifier (GUID)
Handle
Interface Definition Language (IDL)
Interface Identifier (IID)
NetBIOS
Opnum
Phase One
Phase Two

Private Key
Remote Procedure Call (RPC)
Security Identifier (SID)
Server
Symmetric Key
Transaction
Two-Phase Commit
Universally Unique Identifier (UUID)

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

Administration Queue
Connector Application
Connector Queue
Cursor
Dead-Letter Queue
External Transaction
Foreign Queue
Independent Client
Internal Transaction
Message
Microsoft Message Queuing (MSMQ)
Queue
SOAP Reliable Messaging Protocol (SRMP)
Transactional Queue
XML Digital Signature

The following terms are specific to this document:

Format Name: A string that identifies one or more **queues**, as described in [\[MS-MQMQ\]](#) section 2.1.

Internal Certificate Store: A **certificate store** for **server**-owned **certificates**.

Message Queuing System: A distributed system that allows clients to exchange **messages**.
An instance of a **message queuing system** is the **Microsoft Message Queuing (MSMQ)** system.

Negative Source Journaling: The [Queue Manager] retains a copy of each unsuccessfully delivered **message**. Also known as dead-lettering.

Path Name: A string that identifies a **queue**, as specified in [\[MS-MQMQ\]](#) section 2.1.1.

Personal Certificate Store: A **certificate store** for client-owned **certificates**.

Positive Source Journaling: The [Queue Manager] retains a copy of each successfully delivered **message**.

Response Queue: A **queue** via which a client receives application-level response **messages**.

SOAP Attachment: A Multipurpose Internet Mail Extensions (MIME) [\[RFC2045\]](#) binary representation associated with a **SOAP message**.

SOAP Envelope: The wrapping XML element [\[XML\]](#) of a **SOAP message**.

SOAP Message: A **message** that conforms to SOAP protocol [\[SOAP1.1\]](#).

VARIANT: A late-binding (at runtime) data type, structure, or object.

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[C193] The Open Group, "Distributed TP: The XA Specification", February 1992, <http://www.opengroup.org/publications/catalog/c193.htm>

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, <http://www.opengroup.org/public/pubs/catalog/c706.htm>

[FIPS46-3] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 46-3: Data Encryption Standard (DES)", October 1999, <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>

[FIPS186] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 186-2: Digital Signature Standard (DSS)", January 2000, <http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf>

[MC-MQSRM] Microsoft Corporation, "[Message Queuing \(MSMQ\): SOAP Reliable Messaging Protocol \(SRMP\)](#)", October 2007.

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

[MS-COM] Microsoft Corporation, "[Component Object Model Plus \(COM+\) Protocol Specification](#)", March 2007.

[MS-DCOM] Microsoft Corporation, "[Distributed Component Object Model \(DCOM\) Remote Protocol Specification](#)", March 2007.

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

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

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

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

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

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

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

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

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

[MS-OAUT] Microsoft Corporation, "[OLE Automation Protocol Specification](#)", March 2007.

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

[PKCS1] RSA Laboratories, "PKCS#1 Version 2.1: RSA Cryptography Standard", PKCS #1, June 2002, <http://www.rsa.com/rsalabs/node.asp?id=2125>

[RFC1319] Kaliski, B., "The MD2 Message-Digest Algorithm", RFC 1319, April 1992, <http://www.ietf.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.ietf.org/rfc/rfc2119.txt>

[RFC2045] Freed, N., et al., "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996, <http://ietf.org/rfc/rfc2045.txt>

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

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, <http://www.ietf.org/rfc/rfc4122.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>

[SEAL-SPRINGER] Rogaway, P., and Coppersmith, D., "A Software-Optimized Encryption Algorithm", Journal of Cryptography, Springer New York, <http://www.springerlink.com/content/2ydl7vuj2e48wp8/>

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D., "Simple Object Access Protocol (SOAP) 1.1", May 2000, <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

[X509] ITU-T, "Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks", Recommendation X.509, August 2005, <http://www.itu.int/rec/T-REC-X.509/en>

Note There is a charge to download the specification.

[XML] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", W3C Recommendation, September 2006, <http://www.w3.org/TR/REC-xml>

[RFC3174] Eastlake III, D. and Jones, P., "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001, <http://www.ietf.org/rfc/rfc3174.txt>

1.2.2 Informative References

[Box98] D. Box, "Essential COM", Addison-Wesley, 1998, ISBN: 0201634465

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

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

[MSDN-MQCOC] Microsoft Corporation, "Message Queuing COM Components", <http://msdn2.microsoft.com/en-us/library/ms706188.aspx>

1.3 Protocol Overview (Synopsis)

The MSMQ: ActiveX Client Protocol defines how clients interact with a **server** that represents the **message queuing system**.

This document describes the functionality of the message queuing system in an abstract manner; that is, the document describes the client protocol that exercises an abstract system model, rather than a specific implementation. However, some aspects of the client protocol reference specific details of the implementation of the Microsoft message queuing system (known as **Microsoft Message Queuing (MSMQ)**). Where that happens, the abstract meaning of the concepts is described together with informative references to the specific MSMQ implementation details.

The following figure illustrates the abstract concepts and inter-relationships upon which this protocol is defined.

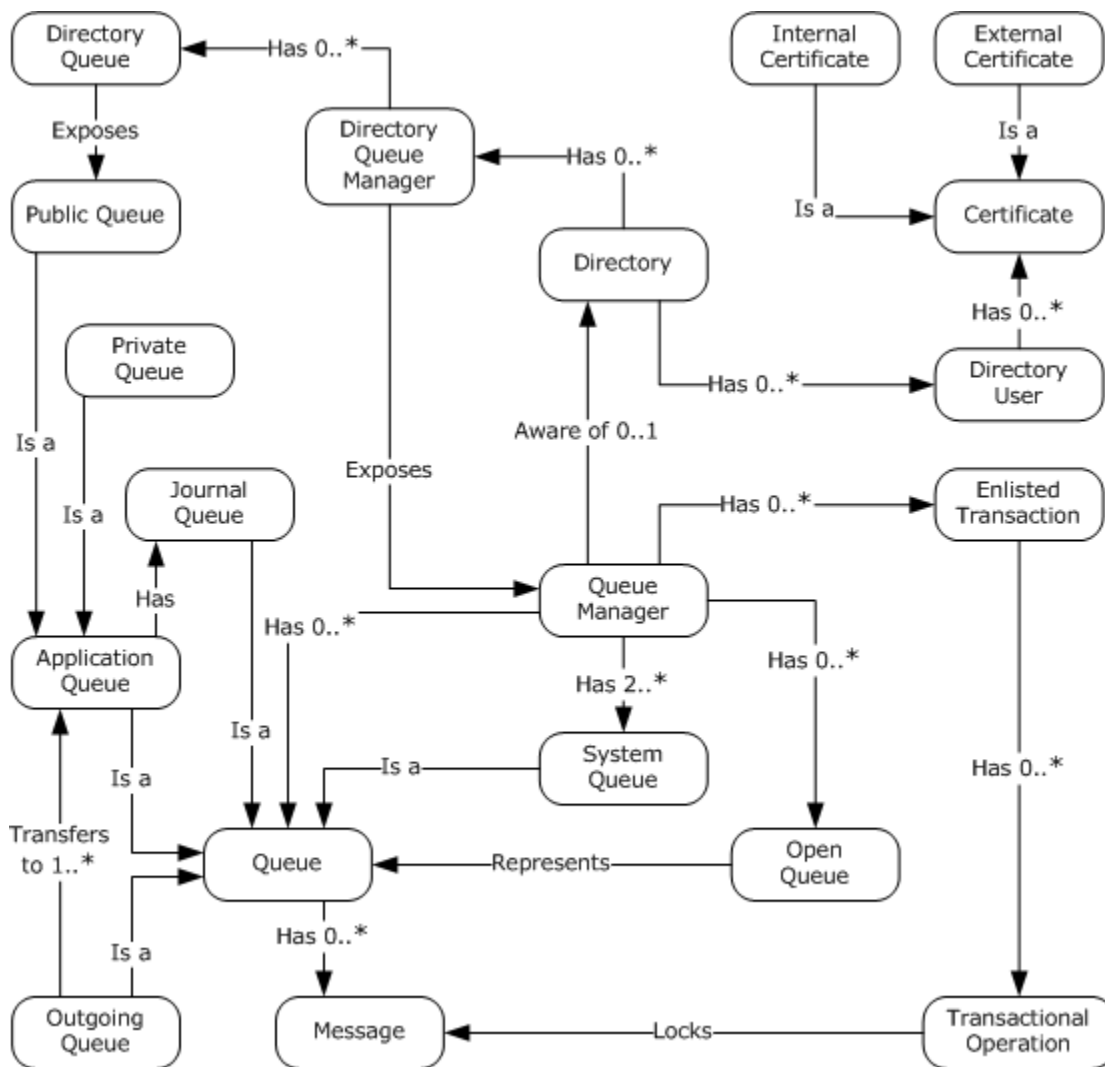


Figure 1: Message queuing system

A message queuing system is defined as consisting of one or more [Queue Manager]s, which facilitate message exchanges between clients of this protocol, and a [Directory] that exposes relevant information about [Queue Manager]s. Clients of this protocol primarily discover information in a [Directory] and operate via [Queue Manager]s. A messaging queuing system implementation is not restricted to any particular distributed system topology, as long as the externally visible behavior adheres to what is described in this document.

A [Queue Manager] has sets of [Queue]s, [Open Queue]s, and [Enlisted Transaction]s.

A [Queue] contains a set of [Message]s. [Message]s are sent and received by clients of this protocol. A [Queue] is owned by exactly one [Queue Manager].

An [Application Queue] is a [Queue] created by, and maintained for the specific purposes of, the clients of this protocol. An [Application Queue] has a [Journal Queue] that is created automatically. The [Journal Queue] contains [Message]s removed from the associated [Application Queue].

A [Public Queue] is an [Application Queue] that is exposed in the [Directory] via the corresponding [Directory Queue]. [Public Queue]s are discoverable by [Directory] lookup. Conversely, a [Private Queue] is an [Application Queue] that is not exposed in any [Directory], and is therefore not discoverable.

When clients send messages using this protocol, they are either deposited directly in the destination [Application Queue] if it is owned by the [Local Queue Manager] or, if it is not local, the messages are kept locally, and asynchronously moved to the destination [Application Queue]s owned by [Queue Manager]s on a remote **computer**. This movement of messages is referred to as the message transfer process. Messages pending transfer are contained in [Outgoing Queue]s, which are [Queue]s owned by the [Local Queue Manager]. Section [1.4.1](#) provides further details on the transfer protocol.

An [Open Queue] represents a [Queue Manager]'s granted rights to clients of this protocol for the purposes of operating on a [Queue].

An [Enlisted Transaction] represents a set of send/receive operations to be performed as an atomic unit of work. A [Transactional Operation] corresponds to send/receive operations that clients of this protocol perform on [Message]s in [Queue]s. A [Transactional Operation] locks the [Message] that the send/receive operation is targeting.

A [Queue Manager] may be affiliated with a [Directory] that exposes the [Queue Manager]. A [Directory] has a set of [Directory User]s and [Directory Queue Manager]s. A [Directory User] contains [Certificate]s that are used to authenticate [Message]s transferred from [Outgoing Queue]s. A [Directory Queue Manager] exposes exactly one [Queue Manager] and its properties. A [Directory Queue Manager] contains a set of [Directory Queue]s, each of which exposes exactly one [Public Queue] and its properties.

In addition to the above descriptions, the message queuing system incorporates the following capabilities:

- Transactions - Messages can be sent and received within the scope of a transaction. Transactions can either be internal, in which case they are coordinated by the message queuing system itself; or distributed, in which case they are controlled by an **external transaction** coordinator.
- Notifications - The message queuing system provides notifications of various events that occur in the system. Such notifications are exposed as internally generated administrative messages that are deposited in client-specified and system-managed **queues**. Administrative messages are distinguished from standard [Message]s through the [Class] property that identifies their meaning. Administrative messages can be categorized as follows: delivery acknowledgments; negative acknowledgments that are used for the asynchronous reporting of error conditions that occur during the message transfer process; and reports about the behavior of the system, known as trace messages.
- Delivery Assurances - The message queuing system supports two delivery assurance modes, Express and Recoverable. Choosing between these modes, through the [Delivery Guarantee] property of the [Message], controls the trade-off between performance (in terms of message throughput), and reliability (in terms of delivery guarantees). Specifically, Recoverable messages survive system shutdowns, and are therefore typically stored in durable storage. Conversely, Express messages do not survive system shutdowns, and therefore don't incur the same costs in terms of durable storage access. Consequently, Express messages typically offer faster communications than Recoverable messages.
- Security - The message queuing system provides the following security-related features: user authentication, queue-level access control over the sending and receiving of messages, and

signing and encryption to ensure integrity and privacy of message exchange during the message transfer process.

- Message Journaling – The message queuing system supports the ability to record copies of messages that pass through the system, a process termed message journaling. Journaling is controlled via the [Journaling Requested] property of the [Message]. If journaling is enabled, when a message is received by a client, and consequently removed from an [Application Queue], then a copy of the [Message] is placed in the [Application Queue]'s [Journal Queue].
- Message expiry – The message queuing system allows clients to specify the useful lifetime of the [Message]s that they send via two expiry properties: [Time To Reach Queue] and [Time To Be Received]. If the relevant conditions are not met within the specified times, then the [Message] is removed from the [Queue] in which it is contained, and moved to the [Queue] identified by the [System Deadletter Queue] property of the [Queue Manager] through which the [Message] was originally sent. This behavior is controlled by the [Journaling Requested] property of the [Message].

Additional details on the concepts summarized above are found in section [3.1.1](#).

1.3.1 Scenario: Retrieving the count of [Messages] in a [Queue]

The following diagram depicts the use of [MSMQManagement](#) to retrieve the count of [Message]s in a [Queue]. First, an instance of the MSMQManagement object is created. Next, the [MSMQManagement.Init](#) method is called to initialize the instance to represent the state of the [Queue] of interest. Finally, the [MSMQManagement.MessageCount](#) method is called, which returns the [Message Count] of the [Messages Table] of the represented [Queue].

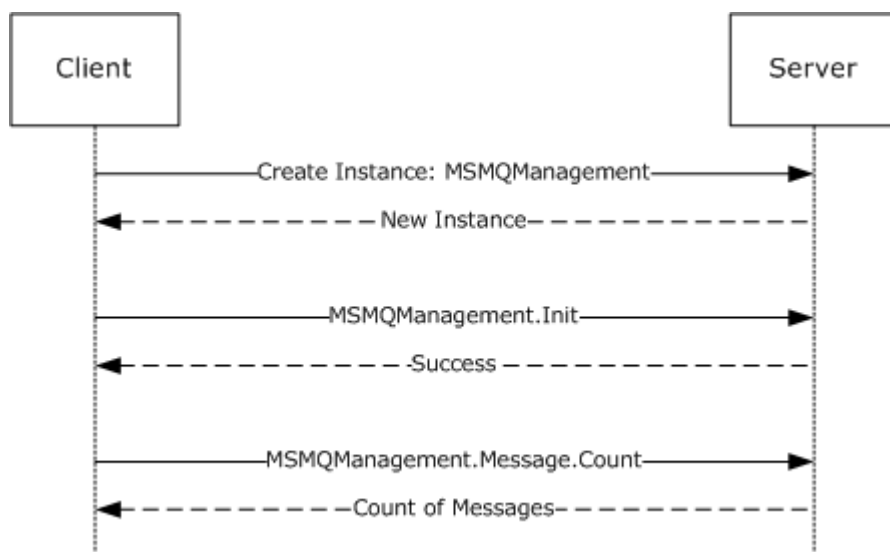


Figure 2: Retrieving the count of [Messages] in a [Queue]

1.3.2 Scenario: Retrieving [EODTargetInfos] for an [Application Queue]

The following diagram depicts the use of [MSMQManagement](#), [MSMQQueueManagement](#), and [MSMQCollection](#) in the retrieval of the EODTargetInfo collection from the [EODTargetInfos] property of an [Application Queue], followed by the retrieval of the SeqNo property, which was arbitrarily selected from the available properties for the purposes of this example. First, an instance of the MSMQManagement object is created. Next, the [MSMQManagement.Init](#) method is called to

initialize the instance to represent the state of the [Application Queue] of interest. To obtain more detailed information about the [Application Queue], the MSMQManagement instance is queried as described in section 3.4.3 to obtain an MSMQQueueManagement instance. Then the [MSMQQueueManagement.EodGetReceiveInfo](#) method is called, which returns an MSMQCollection instance which contains the EODTargetInfo collection for the [EODTargetInfos] property of the [Application Queue] represented by the MSMQQueueManagement instance. Finally, the [MSMQCollection.Item](#) method is called, specifying the SeqNo property.

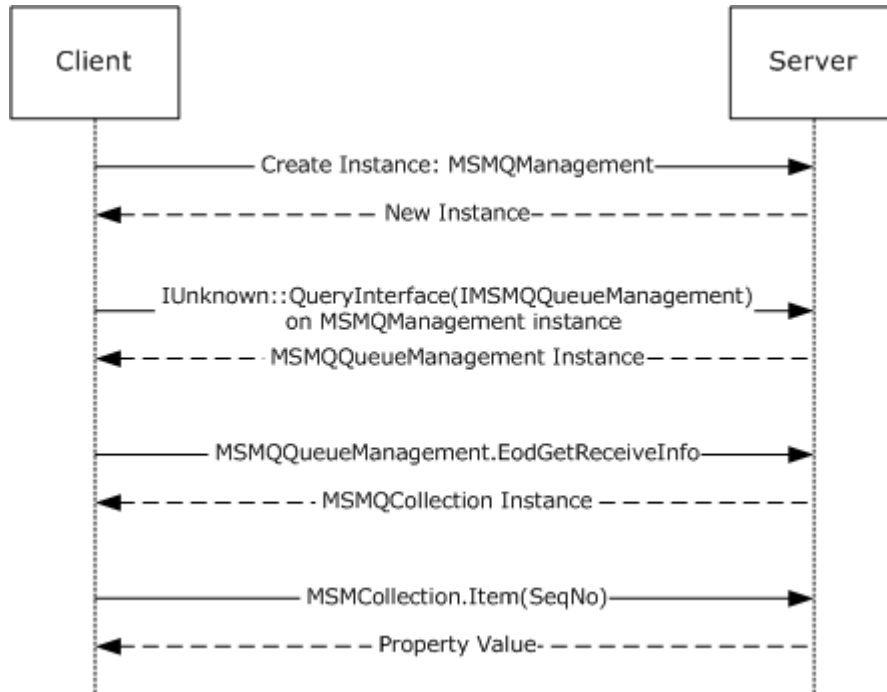


Figure 3: Retrieving [EODTargetInfos] for an [Application Queue]

1.3.3 Scenario: Pausing an [Outgoing Queue]

The following diagram depicts the use of [MSMQManagement](#) and [MSMQOutgoingQueueManagement](#) to pause transmission of messages from an [Outgoing Queue]. First, an instance of the MSMQManagement object is created. Next, the [MSMQManagement.Init](#) method is called to initialize the instance to represent the state of the [Outgoing Queue] of interest. To obtain functionality specific to [Outgoing Queue]s, the MSMQManagement instance is queried as described in section 3.5.3 to obtain an MSMQOutgoingQueueManagement instance. Finally, the [MSMQOutgoingQueueManagement.Pause](#) method is called.

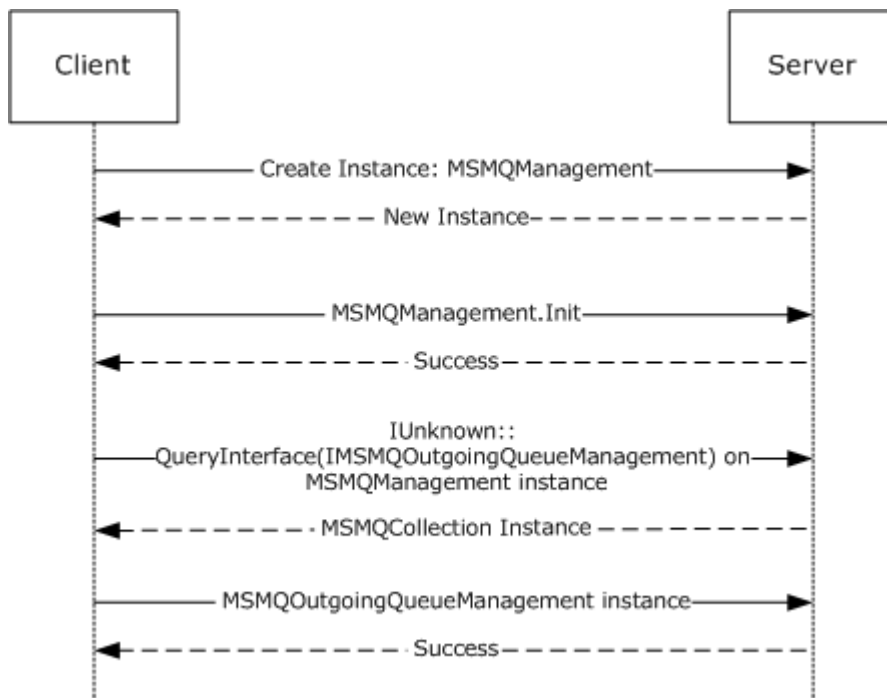


Figure 4: Pausing an [Outgoing Queue]

1.3.4 Scenario: Discovering [Directory Queue]s in the [Directory]

The following diagram depicts the use of [MSMQQuery](#), [MSMQQueueInfos](#), and [MSMQQueueInfo](#) to discover [Directory Queue]s in the [Directory] according to some search criteria. First, an instance of the MSMQQuery object is created. Next, the [MSMQQuery.LookupQueue](#) method is called with some search criteria, which returns an MSMQQueueInfos instance containing information on all [Queue]s in the [Directory] which meet the given criteria. To obtain that information, the [MSMQQueueInfos.Next](#) method is called, which returns an MSMQQueueInfo instance containing information on the first [Directory Queue] discovered. That is optionally followed by additional calls to the **MSMQQueueInfos.Next** method, each of which returns another MSMQQueueInfo instance containing information about another discovered [Directory Queue].

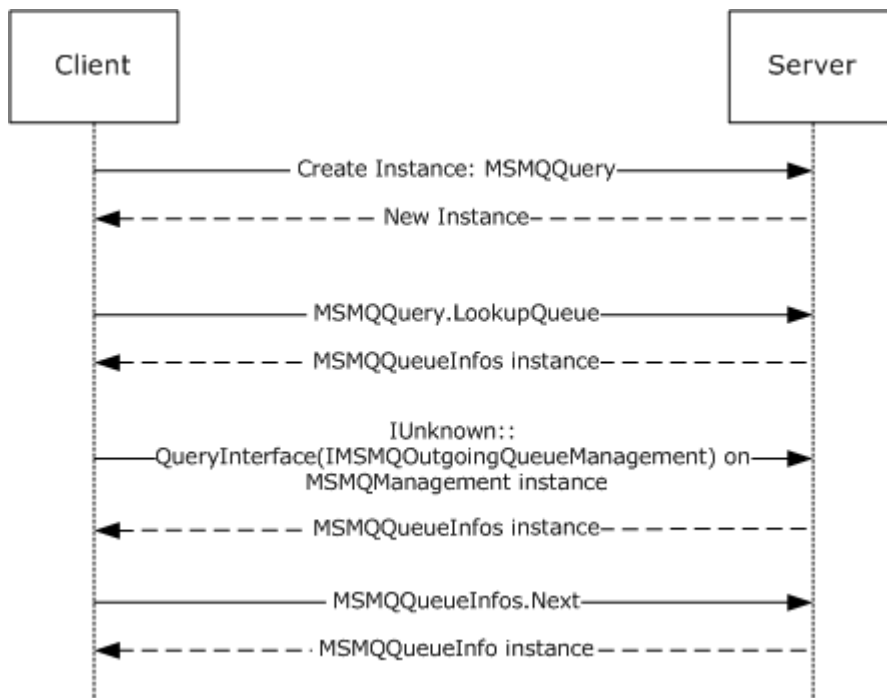


Figure 5: Discovering [Directory Queue]s in the [Directory]

1.3.5 Scenario: Receiving a message from a queue asynchronously via event callbacks

The following diagram depicts the use of [MSMQQueueInfo](#), [MSMQQueue](#), and [MSMQEvent](#) to receive a message from a queue asynchronously using event callbacks. To start with, a [MSMQQueueInfo](#) object instance which references a particular [Queue] is required. There are various ways to achieve this; for the purposes of this example, an instance is created and its `FormatName` property is set. Then the [MSMQQueueInfo.Open](#) method is called to open the referenced [Queue] for receive access, which returns a [MSMQQueue](#) instance representing the [Open Queue]. Next, an [MSMQEvent](#) object instance is created. The [MSMQEvent](#) instance is passed to the [MSMQQueue.EnableNotification](#) method. When a message arrives, the server calls the [MSMQEvent.Arrived](#) method. In response to that call, the [MSMQQueue.Receive](#) method is called, which returns an [MSMQMessage](#) instance.

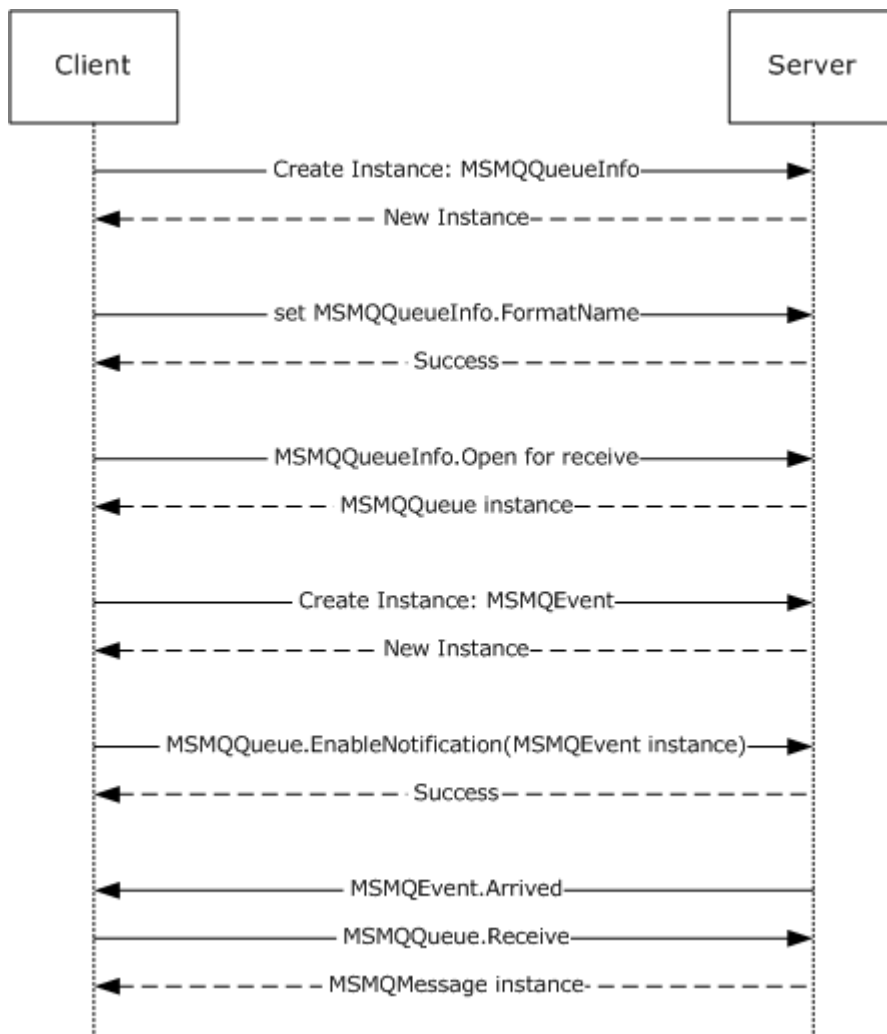


Figure 6: Receiving a message from a queue asynchronously via event callbacks

1.3.6 Scenario: Sending a multicast message

The following diagram depicts the use of [MSMQMessage](#), and [MSMQDestination](#) to send a multicast message. First, an instance of the [MSMQMessage](#) is created and its properties set. Next, an instance of [MSMQDestination](#) is created and the format name is set. Next, the [MSMQMessage.Send](#) method is called to send the message. Finally the [MSMQDestination](#) is closed.

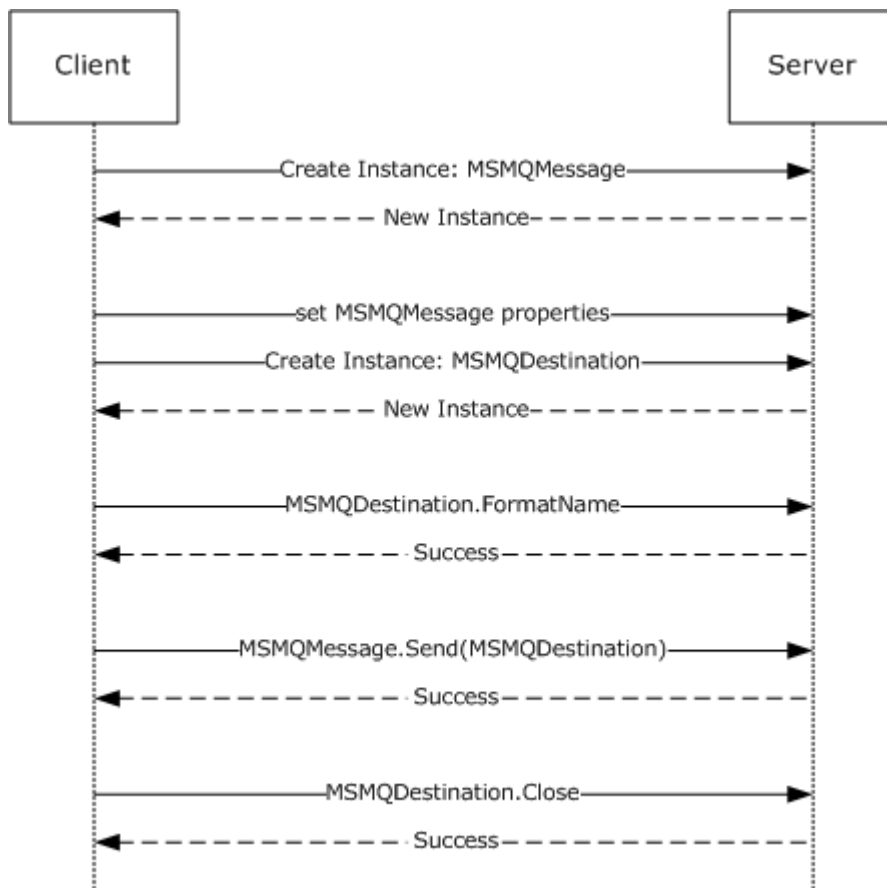


Figure 7: Sending a multicast message

1.3.7 Scenario: Sending a message with an internal transaction

The following diagram depicts the use of [MSMQQueue](#), [MSMQQueueInfo](#), [MSMQMessage](#), [MSMQCoordinatedTransactionDispenser](#), and [MSMQTransaction](#) to send a message to a queue with an internal transaction. First, an instance of [MSMQMessage](#) is created and its properties are set. Next an instance of [MSMQQueueInfo](#) is created and the pathname is set. Next, the [MSMQQueueInfo.Open](#) method is called to get an instance of the [MSMQQueue](#). Next, an instance of the [MSMQCoordinatedTransactionDispenser](#) is created. The method [MSMQCoordinatedTransactionDispenser.BeginTransaction](#) is called to get an instance of [MSMQTransaction](#). Next, the [MSMQMessage.Send](#) method is called to send the message to the queue. Next the transaction is committed by calling [MSMQTransaction.Commit](#). Finally the [MSMQQueue](#) is closed.

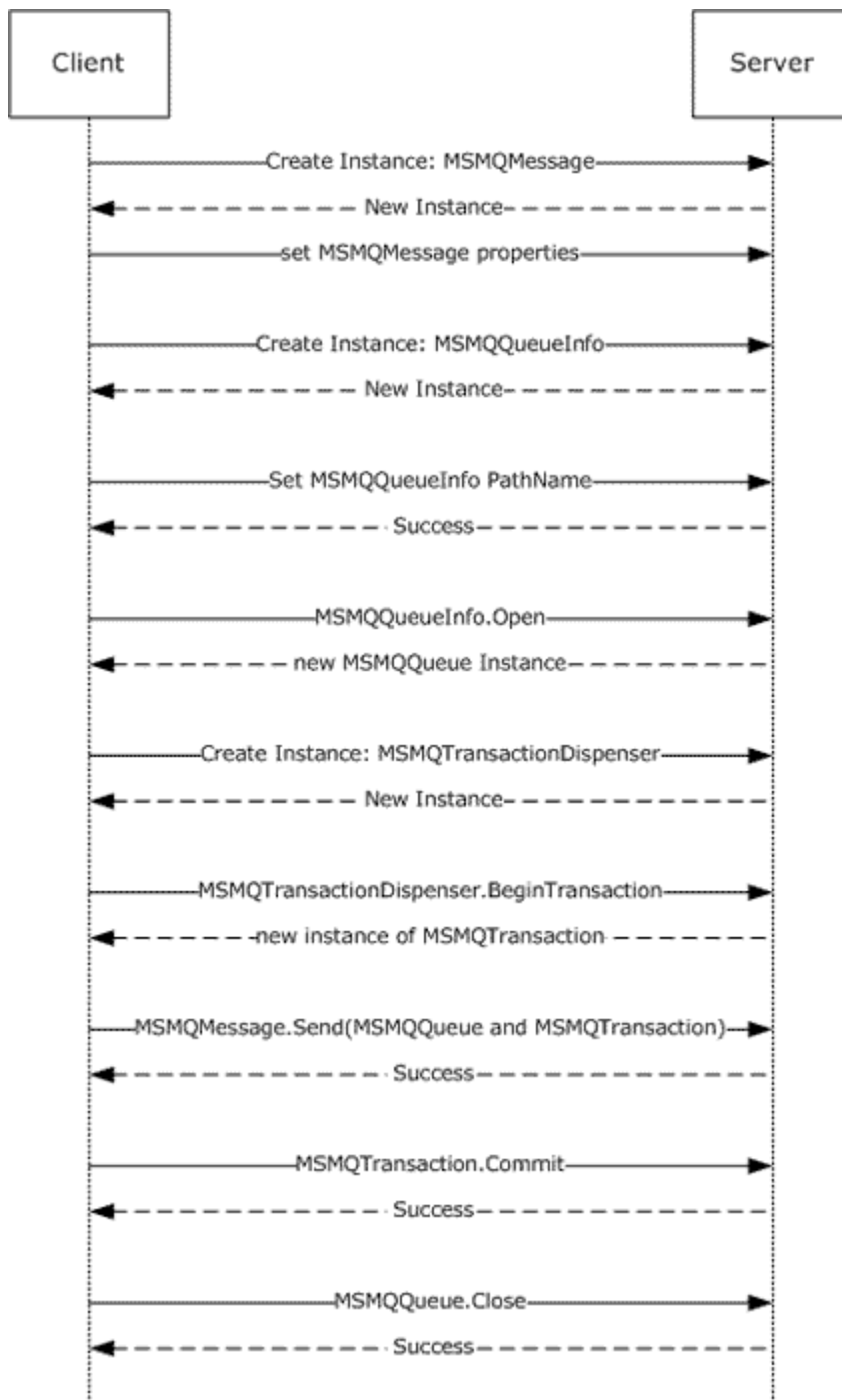


Figure 8: Sending a message with an internal transaction

1.3.8 Scenario: Sending a message with an external transaction

The following diagram depicts the use of [MSMQQueue](#), [MSMQQueueInfo](#), [MSMQMessage](#), [MSMQCoordinatedTransactionDispenser](#), and [MSMQTransaction](#) to send a message to a queue with

an external transaction. First, an instance of the MSMQMessage is created and its properties set. Next an instance of MSMQQueueInfo is created and the pathname is set. Next, the [MSMQQueueInfo.Open](#) method is called to get an instance of the MSMQQueue. Next, an instance of the MSMQCoordinatedTransactionDispenser is created. The method [MSMQCoordinatedTransactionDispenser.BeginTransaction](#) is called to get an instance of MSMQTransaction. Next, the [MSMQMessage.Send](#) method is called to send the message to the queue. Next the transaction is committed by calling [MSMQTransaction.Commit](#). Finally the MSMQQueue is closed.

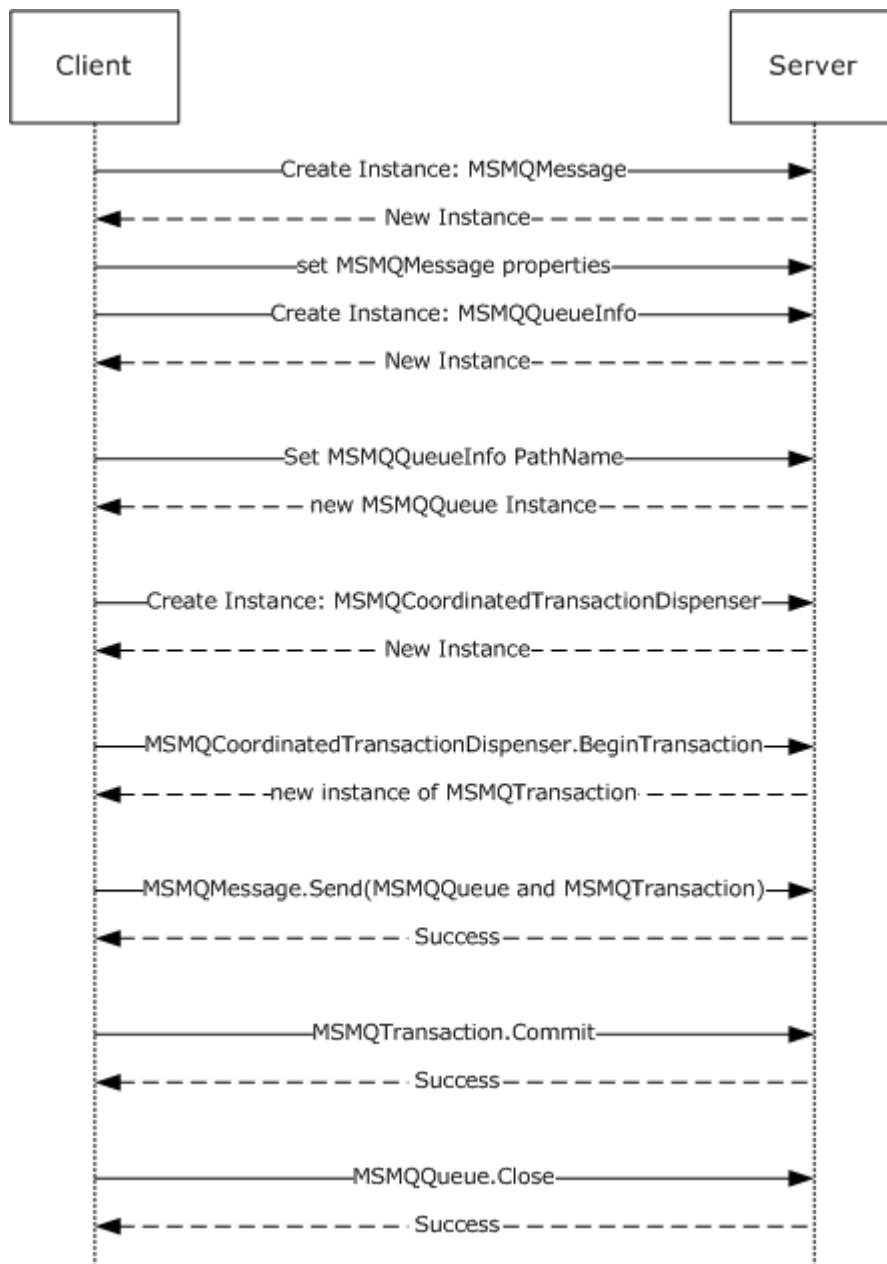


Figure 9: Sending a message with an external transaction

1.3.9 Scenario: Sending [Message] to a [Queue]

The following diagram depicts the use of [MSMQQueue](#), [MSMQQueueInfo](#), and [MSMQMessage](#) to send a [Message] to a [Queue]. First, an instance of the MSMQMessage is created and its properties set. Next an instance of MSMQQueueInfo is created and the format name is set. Next, the [MSMQQueueInfo.Open](#) method is called to get an instance of the MSMQQueue. Finally, the [MSMQMessage.Send](#) method is called to send the [Message] to a [Queue].

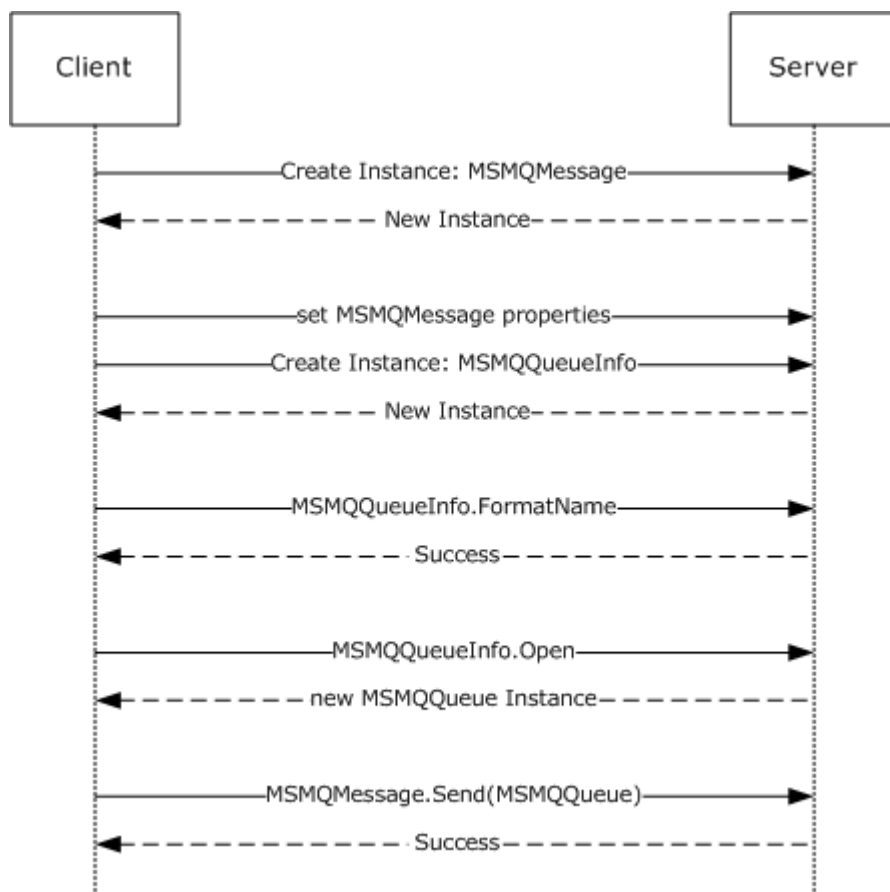


Figure 10: Sending [Message] to a [Queue]

1.4 Relationship to Other Protocols

The MSMQ: ActiveX Client Protocol defines a comprehensive object model for clients to utilize the services of a message queuing system. As such, many methods defined by this protocol enable the client to configure or manipulate aspects of the server that are defined in detail by other protocol specifications. This specification does not mandate specific protocols for such instances, but it does reference specific exemplary protocol specifications when they describe the aspects of the server that this protocol configures or manipulates.

The protocols described in the remainder of this section are referenced throughout this document to describe external protocols with which an implementation of the MSMQ: ActiveX Client Protocol must interact.

1.4.1 Message Transfer

As mentioned in the Protocol Overview, a typical message queuing system is implemented by a number of distributed [Queue Manager]s that collaborate to move messages to the appropriate [Queue]s. The message transfer process of moving messages from an [Outgoing Queue] owned by one [Queue Manager] to one or more [Queue]s owned by another [Queue Manager] is described abstractly in this document, since different message queuing system architectures may realize this concept in different ways.

The MSMQ message queuing system implements the message transfer process using a direct [Queue Manager] to [Queue Manager] binary protocol that is described in full in [\[MS-MQQB\]](#).

1.4.2 Directory Access

The MSMQ: ActiveX Client Protocol describes a server implementation that may require the ability to create, delete, update, and query records that are managed by a **directory service**. Server access to a directory service is an optional configuration.

This specification does not mandate the specific protocol that is used to communicate with a directory service. Suitable protocols are described in [\[MS-MQDS\]](#) and [\[MS-ADTS\]](#).

An abstract data model for the directory that is mandated by this protocol is specified in sections [3.1.1.1](#) through [3.1.1.11](#).

1.4.3 Distributed Transaction Coordination

The MSMQ: ActiveX Client Protocol describes a server implementation that interacts with a distributed transaction coordinator. This specification does not mandate the specific protocol that is used to communicate with the distributed transaction coordinator. A suitable protocol is described in [\[MS-DTCO\]](#).

1.5 Prerequisites/Preconditions

This protocol is implemented over DCOM and **RPC** and, as a result, has the prerequisites identified in [\[MS-DCOM\]](#), [\[MS-OAUT\]](#), [\[MS-COM\]](#), and [\[MS-RPCE\]](#) as being common to the Distributed Component Object Model, OLE Automation, Component Object Model Plus, and Remote Procedure Call Extensions interfaces. See the following protocol stack diagram:

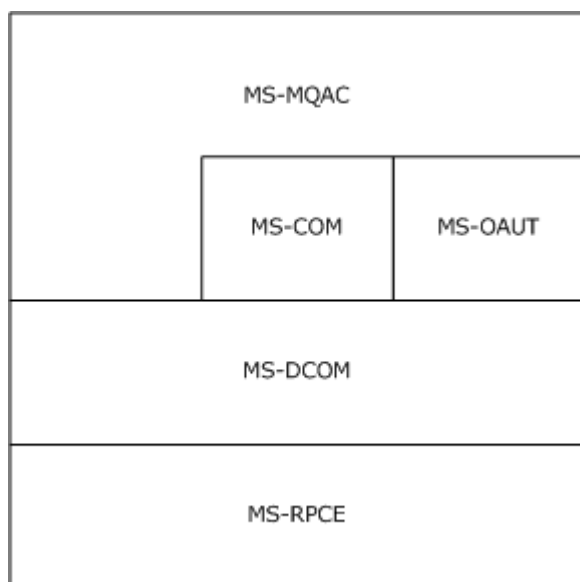


Figure 11: Protocol stack

The MSMQ: ActiveX Client Protocol assumes that a client has obtained the name of a server that supports this protocol suite before the protocol is invoked. How a client obtains this server name is outside the purview of this specification (see [\[MS-MQMA\]](#) section 2.2 for information about client profiles). The protocol also assumes that the client has sufficient security privileges to interact with the message queuing system.

1.6 Applicability Statement

The MSMQ: ActiveX Client Protocol provides clients with remote access to the functionality of the message queuing system. Message queuing applications on client computers use this protocol to instruct the server to perform message queuing operations in response to client requests, as if the client application was executing locally on the server computer.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- Supported Transports: The protocol uses the [DCOM Remote Protocol](#), which in turn uses RPC over TCP, as its only transport. See section [1.5](#) for details.
- Protocol Version: This protocol is comprised of several DCOM classes, which are described in section [3](#). Each class MAY feature multiple interface revisions, where each interface revision is a binary-compatible successor to the previous revision.

Capability Negotiation: The client negotiates for a given set of server functionalities when binding to the server, by specifying the **UUID** that corresponds to the desired RPC interface via the COM [IUnknown::QueryInterface](#) method (for more information, see section [3.1](#)).

1.8 Vendor-Extensible Fields

This protocol uses 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.

1.9 Standards Assignments

The following table contains well-known **GUIDs** that are defined by the MSMQ: ActiveX Client Protocol. Included are **CLSID** and **interface identifier (IID)** GUIDs.

Parameter	Value
RPC interface UUID for IMSMQQuery	d7d6e072-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQQuery2	eba96b0e-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQQuery3	eba96b19-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQQuery4	eba96b24-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQQuery	d7d6e073-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQMessage	d7d6e074-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQMessage2	d9933be0-a567-11d2-b0f3-00e02c074f6b
RPC interface UUID for IMSMQMessage3	eba96b1a-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQMessage4	eba96b23-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQMessage	d7d6e075-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQQueue	d7d6e076-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQQueue2	ef0574e0-06d8-11d3-b100-00e02c074f6b
RPC interface UUID for IMSMQQueue3	eba96b1b-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQQueue4	eba96b20-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQQueue	d7d6e079-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQPrivateEvent	d7ab3341-c9d3-11d1-bb47-0080c7c5a2c0
RPC interface UUID for IMSMQEvent	d7d6e077-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQEvent2	eba96b12-2168-11d3-898c-

Parameter	Value
	00e02c074f6b
RPC interface UUID for IMSMQEvent3	eba96b1c-2168-11d3-898c-00e02c074f6b
RPC interface UUID for DMSMQEventEvents	d7d6e078-dccd-11d0-aa4b-0060970debae
Coclass UUID for MSMQEvent	d7d6e07a-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQQueueInfo	d7d6e07b-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQQueueInfo2	fd174a80-89cf-11d2-b0f2-00e02c074f6b
RPC interface UUID for IMSMQQueueInfo3	eba96b1d-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQQueueInfo4	eba96b21-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQQueueInfo	d7d6e07c-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQQueueInfos	d7d6e07d-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQQueueInfos2	eba96b0f-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQQueueInfos3	eba96b1e-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQQueueInfos4	eba96b22-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQQueueInfos	d7d6e07e-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQTransaction	d7d6e07f-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQTransaction2	2ce0c5b0-6e67-11d2-b0e6-00e02c074f6b
RPC interface UUID for IMSMQTransaction3	eba96b13-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQTransaction	d7d6e080-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQCoordinatedTransactionDispenser	d7d6e081-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQCoordinatedTransactionDispenser2	eba96b10-2168-11d3-898c-00e02c074f6b

Parameter	Value
RPC interface UUID for IMSMQCoordinatedTransactionDispenser3	eba96b14-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQCoordinatedTransactionDispenser	d7d6e082-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQTransactionDispenser	d7d6e083-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQTransactionDispenser2	eba96b11-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQTransactionDispenser3	eba96b15-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQTransactionDispenser	d7d6e084-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQApplication	d7d6e085-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQApplication2	12a30900-7300-11d2-b0e6-00e02c074f6b
RPC interface UUID for IMSMQApplication3	eba96b1f-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQApplication	d7d6e086-dccd-11d0-aa4b-0060970debae
RPC interface UUID for IMSMQDestination	eba96b16-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQPrivateDestination	eba96b17-2168-11d3-898c-00e02c074f6b
Coclass UUID for MSMQDestination	eba96b18-2168-11d3-898c-00e02c074f6b
RPC interface UUID for IMSMQCollection	0188ac2f-ecb3-4173-9779-635ca2039c72
Coclass UUID for MSMQCollection	f72b9031-2f0c-43e8-924e-e6052cdc493f
RPC interface UUID for IMSMQManagement	be5f0241-e489-4957-8cc4-a452fcf3e23e
Coclass UUID for MSMQManagement	39ce96fe-f4c5-4484-a143-4c2d5d324229
RPC interface UUID for IMSMQOutgoingQueueManagement	64c478fb-f9b0-4695-8a7f-439ac94326d3
Coclass UUID for MSMQOutgoingQueueManagement	0188401c-247a-4fed-99c6-bf14119d7055
RPC interface UUID for IMSMQQueueManagement	7fbe7759-5760-444d-b8a5-

Parameter	Value
	5e7ab9a84cce
Coclass UUID for MSMQQueueManagement	33b6d07e-f27d-42fa-b2d7-bf82e11e9374

2 Messages

2.1 Transport

This protocol is implemented as a collection of DCOM interfaces, and as a result, completely inherits the transport requirements specified in [\[MS-DCOM\]](#) section 2.1.

2.2 Common Data Types

In addition to RPC base types and definitions specified in [\[C706\]](#) and [\[MS-RPCE\]](#), additional data types are defined in this section.

2.2.1 OLE Automation Data Types

Refer to [\[MS-OAUT\]](#) section 2.2 for descriptions of the following data types:

- BSTR
- DATE
- IID
- REFIID
- VARIANT
- VARIANT_BOOL

2.2.2 Enumerations

The following enumerated types are defined below:

- [MQACCESS](#)
- [MQAUTHENTICATE](#)
- [MQCLAG](#)
- [MQJOURNAL](#)
- [MQMSGACKNOWLEDGEMENT](#)
- [MQMSGAUTHENTICATION](#)
- [MQMSGAUTHLEVEL](#)
- [MQMSGCLASS](#)
- [MQMSGCURSOR](#)
- [MQMSGDELIVERY](#)
- [MQMSGJOURNAL](#)
- [MQMSGPRIVLEVEL](#)

- [MQMSGSENDERIDTYPE](#)
- [MQMSGTRACE](#)
- [MQPRIVLEVEL](#)
- [MQSHARE](#)
- [MQTRANSACTION](#)
- [MQTRANSACTIONAL](#)
- [QUEUE_STATE](#)
- [RELOPS](#)
- [XACTTC](#)

2.2.2.1 MQTRANSACTION

The **MQTRANSACTION** enumeration defines values passed to methods that occur in the scope of a transaction. Specific values in this enumeration indicate the manner in which an operation obtains a transaction identifier. A value is reserved to indicate that the operation is not to be performed in the scope of a transaction.

```
typedef enum
{
    MQ_NO_TRANSACTION = 0x00000000,
    MQ_MTS_TRANSACTION = 0x00000001,
    MQ_XA_TRANSACTION = 0x00000002,
    MQ_SINGLE_MESSAGE = 0x00000003
} MQTRANSACTION;
```

MQ_NO_TRANSACTION: The operation is not to be performed in the context of a transaction. Therefore, the operation is to be immediately executed to completion.

MQ_MTS_TRANSACTION: The operation MUST be performed in the context of an MSDTC coordinated transaction [\[MS-DTCO\]](#). The transaction identifier is to be obtained from the Component Object Model (COM) activation context, as defined in [\[MS-COM\]](#).

MQ_XA_TRANSACTION: The operation MUST be performed in the context of an XA coordinated transaction, as defined by [\[C193\]](#).

The transaction identifier is to be obtained from the COM activation context, as defined in [\[MS-COM\]](#).

MQ_SINGLE_MESSAGE: The operation is not to be performed in the context of a transaction. This flag is required when sending to, or receiving a single message from, a **transactional queue** when not coordinating a transaction with systems beyond message queuing.

Used by:

- [IMSMQQueue::Receive](#)
- [IMSMQQueue::Receive_v1](#)
- [IMSMQQueue::ReceiveByLookupId](#)

- [IMSMQQueue::ReceiveCurrent](#)
- [IMSMQQueue::ReceiveCurrent_v1](#)
- [IMSMQQueue::ReceiveFirstByLookupId](#)
- [IMSMQQueue::ReceiveLastByLookupId](#)
- [IMSMQQueue::ReceiveNextByLookupId](#)
- [IMSMQQueue::ReceivePreviousByLookupId](#)
- [IMSMQQueue::ReceiveByLookupIdAllowPeek](#)
- [IMSMQMessage::Send](#)

2.2.2.2 MQSHARE

The **MQSHARE** enumeration defines values that indicate the requested exclusivity level when opening a [Queue]. When a queue is opened for exclusive access, other clients of this protocol MUST NOT be permitted to open the same queue. Furthermore, a client of this protocol MUST NOT acquire exclusive access to a queue that is already opened by another client.

```
typedef enum
{
    MQ_DENY_NONE = 0x00000000,
    MQ_DENY_RECEIVE_SHARE = 0x00000001
} MQSHARE;
```

MQ_DENY_NONE: This value indicates that the [Queue] is to be opened for non-exclusive access. If any [Open Queue] that refers to the [Queue] has an [IsExclusive] value of True, the [IMSMQQueueInfo4::Open](#) method MUST return an error HRESULT. The [IsExclusive] property of the [Open Queue] that was created by the **Open** method MUST be set to False.

MQ_DENY_RECEIVE_SHARE: The [Queue] MUST be opened for exclusive access. If any [Open Queue] already refers to the [Queue], the **Open** method MUST return an error HRESULT. If the [Queue] is opened successfully, the [IsExclusive] property of the [Open Queue] MUST be set to True.

Note the difference between **MQSHARE** and [MQACCESS](#):

- **MQSHARE** specifies whether a client has exclusive access to a queue, thereby making the queue inaccessible to other clients.
- **MQACCESS** determines the access to messages within a queue, such as read-only or read-write access to messages.

Used by:

- **IMSMQQueueInfo::Open**

2.2.2.3 MQACCESS

The **MQACCESS** enumeration defines values that indicate the requested access mode when opening a [Queue]. When a queue is opened for exclusive access, other clients of this protocol MUST NOT be

permitted to open the same queue. Furthermore, a client of this protocol MUST NOT acquire exclusive access to a queue that is already opened by any other client.

```
typedef enum
{
    MQ_RECEIVE_ACCESS = 0x00000001,
    MQ_SEND_ACCESS = 0x00000002,
    MQ_PEEK_ACCESS = 0x00000020,
    MQ_ADMIN_ACCESS = 0x00000080
} MQACCESS;
```

MQ_RECEIVE_ACCESS: The [Open Queue] created by the [IMSMQQueueInfo4::Open](#) method represents permission granted by the [Queue Manager] to read and delete [Message]s from the [Messages Table] contained by the [Queue] that is referenced by the [Open Queue].

MQ_SEND_ACCESS: The [Open Queue] created by the **Open** method represents permission granted by the [Queue Manager] to insert new [Message]s into the [Messages Table] contained by the [Queue] that is referenced by the [Open Queue]. This value is not valid when combined with MQ_ADMIN_ACCESS.

MQ_PEEK_ACCESS: The [Open Queue] created by the **Open** method represents permission granted by the [Queue Manager] to read (but not delete) [Message]s from the [Messages Table] contained by the [Queue] that is referenced by the [Open Queue].

MQ_ADMIN_ACCESS: The MQ_ADMIN_ACCESS bit modifies the interpretation of the **format name** by the **Open** method. When specified, this value indicates that the [Outgoing Queue] that transfers to the [Application Queue] identified by the given format name is to be opened, rather than the [Application Queue] itself.

Note the difference between [MQSHARE](#) and **MQACCESS**:

- **MQACCESS** determines the access to messages within a queue, such as read-only or read-write access to messages.
- **MQSHARE** specifies whether a client has exclusive access to a queue, thereby making the queue inaccessible to other clients.

Used by:

- **IMSMQQueueInfo::Open**

2.2.2.4 MQJOURNAL

The **MQJOURNAL** enumeration defines values that indicate the requested target journaling mode for an [Application Queue]. Target journaling is the process of retaining copies of the [Message]s removed from an [Application Queue]. For [Application Queue]s where target journaling is enabled, [Message]s removed from an [Application Queue] are copied to the [Messages Table] of the associated [Journal Queue].

```
typedef enum
{
    MQ_JOURNAL_NONE = 0x00000000,
    MQ_JOURNAL = 0x00000001
} MQJOURNAL;
```

MQ_JOURNAL_NONE: Target journaling is not enabled for the [Application Queue]. [Message]s received from the [Application Queue] by clients of this protocol are permanently deleted.

MQ_JOURNAL: Target journaling is enabled for the [Application Queue]. [Message]s received from the [Application Queue] by clients of this protocol are copied to the [Messages Table] of the associated [Journal Queue] prior to being deleted from the [Application Queue].

Used by:

- [IMSMQQueueInfo::Journal](#)
- [IMSMQQueueInfo::Journal](#)

2.2.2.5 MQTRANSACTIONAL

The **MQTRANSACTIONAL** enumeration defines values that indicate if a [Queue] is a transactional queue.

```
typedef enum
{
    MQ_TRANSACTIONAL_NONE = 0x00000000,
    MQ_TRANSACTIONAL = 0x00000001
} MQTRANSACTIONAL;
```

MQ_TRANSACTIONAL_NONE: The [Queue] is not a transactional queue. The [IsTransactional] element is false.

MQ_TRANSACTIONAL: The [Queue] is a transactional queue. The [IsTransactional] element is True.

Used by:

- get [IMSMQQueueInfo::IsTransactional](#)
- put [IMSMQQueueInfo::IsTransactional](#)

2.2.2.6 MQAUTHENTICATE

The **MQAUTHENTICATE** enumeration defines values that indicate whether an [Application Queue] accepts only authenticated messages. Authenticated messages are [Message]s for which the [Authentication Level] value indicates that the [Message] was signed.

```
typedef enum
{
    MQ_AUTHENTICATE_NONE = 0x00000000,
    MQ_AUTHENTICATE = 0x00000001
} MQAUTHENTICATE;
```

MQ_AUTHENTICATE_NONE: The [Authentication Required] field of the specified [Application Queue] equals False.

The [Application Queue] does not restrict messages on the basis of their authentication status.

MQ_AUTHENTICATE: The [Authentication Required] field of the specified [Application Queue] equals True.

The [Application Queue] accepts only [Message]s for which the [Signature] has been successfully validated upon arrival to the [Application Queue], when delivered via the message transfer process.

Used by:

- get [IMSMQQueueInfo::Authenticate](#)
- put [IMSMQQueueInfo::Authenticate](#)

2.2.2.7 MQPRIVLEVEL

The **MQPRIVLEVEL** enumeration defines values that indicate whether an [Application Queue] accepts only encrypted messages. Encrypted messages are [Message]s for which the [Privacy Level] value indicates that the message was encrypted during the message transfer process.

```
typedef enum
{
    MQ_PRIV_LEVEL_NONE = 0x00000000,
    MQ_PRIV_LEVEL_OPTIONAL = 0x00000001,
    MQ_PRIV_LEVEL_BODY = 0x00000002
} MQPRIVLEVEL;
```

MQ_PRIV_LEVEL_NONE: The [Application Queue] accepts only [Message]s for which the [Privacy Level] value (as defined by the [MQMSGPRIVLEVEL \(section 2.2.2.15\)](#) enumeration) indicates that the message was NOT encrypted during the message transfer process.

MQ_PRIV_LEVEL_OPTIONAL: The [Application Queue] does not restrict [Message]s according to their [Privacy Level] value.

MQ_PRIV_LEVEL_BODY: The [Application Queue] accepts only [Message]s for which the [Privacy Level] value (as defined by the **MQMSGPRIVLEVEL** enumeration) indicates that the message was encrypted during the message transfer process.

Used by:

- get [IMSMQQueueInfo::PrivLevel](#)
- put [IMSMQQueueInfo::PrivLevel](#)

2.2.2.8 MQMSGCURSOR

The **MQMSGCURSOR** enumeration defines values that indicate the **cursor** behavior mode for notifications from an [MSMQQueue](#) method. The cursor behavior mode indicates how the state of the cursor that is associated with an event notification is updated.

```
typedef enum
{
    MQMSG_FIRST = 0x00000000,
    MQMSG_CURRENT = 0x00000001,
    MQMSG_NEXT = 0x00000002
}
```



```
} MQMSGCURSOR;
```

MQMSG_FIRST: The client is notified when a [Message] is available at the head of the [Queue].

The behavior for this value is defined for the [MSMQQueue::EnableNotification](#) method in section [3.11.4.1.9](#).

MQMSG_CURRENT: The client is notified when a [Message] is available at the current cursor position within the [Queue].

The behavior for this value is defined for the **MSMQQueue::EnableNotification** method in section [3.11.4.1.9](#).

MQMSG_NEXT: The cursor is advanced, and the client is notified when a [Message] is available at the advanced cursor position within the [Queue].

The behavior for this value is defined for the **MSMQQueue::EnableNotification** method in section [3.11.4.1.9](#).

Used by:

- **IMSMQQueue::EnableNotification**
- [DMSMQEventEvents::Arrived](#)
- [DMSMQEventEvents::ArrivedError](#)

2.2.2.9 MQMSGCLASS

The **MQMSGCLASS** enumeration defines values that indicate the classification of a [Message]. A [Message] can originate from a client of this protocol, or it can be generated by the operations pertaining to the message transfer process. The values defined for this enumeration indicate the reason that the [Message] was generated.

```
typedef enum
{
    MQMSG_CLASS_NORMAL = 0x0000,
    MQMSG_CLASS_REPORT = 0x0001,
    MQMSG_CLASS_ACK_REACH_QUEUE = 0x0002,
    MQMSG_CLASS_ACK_RECEIVE = 0x4000,
    MQMSG_CLASS_NACK_BAD_DST_Q = 0x8000,
    MQMSG_CLASS_NACK_PURGED = 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_COULD_NOT_ENCRYPT = 0x8008,
    MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_Q = 0x8009,
    MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_MSG = 0x800a,
    MQMSG_CLASS_NACK_UNSUPPORTED_CRYPTO_PROVIDER = 0x800b,
    MQMSG_CLASS_NACK_SOURCE_COMPUTER_GUID_CHANGED = 0x800c,
    MQMSG_CLASS_NACK_Q_DELETED = 0xc000,
    MQMSG_CLASS_NACK_Q_PURGED = 0xc001,
    MQMSG_CLASS_NACK_RECEIVE_TIMEOUT = 0xc002,
```

```
MQMSG_CLASS_NACK_RECEIVE_TIMEOUT_AT_SENDER = 0xc003  
} MQMSGCLASS;
```

MQMSG_CLASS_NORMAL: The message originated from a client of this protocol via a call to [MSMQMessage4::Send](#).

MQMSG_CLASS_REPORT: The message was generated by the route tracing feature of the message transfer process. Messages of type MQMSG_CLASS_REPORT are generated while [Message]s for which [Tracing Requested] is True arrive at [Queue]s along the route to the final destination.

MQMSG_CLASS_ACK_REACH_QUEUE: The message was generated as a result of a [Message] successfully arriving at its destination [Application Queue].

MQMSG_CLASS_ACK_RECEIVE: The message was generated as a result of a [Message] being successfully retrieved by a client of this protocol.

MQMSG_CLASS_NACK_BAD_DST_Q: The message was generated to indicate that delivery of the [Message] was cancelled because the destination [Application Queue] was unreachable.

MQMSG_CLASS_NACK_PURGED: The message was generated to indicate that the [Message] was deleted prior to arriving at the destination [Application Queue].

MQMSG_CLASS_NACK_REACH_QUEUE_TIMEOUT: The message was generated to indicate that the [Message].[Time To Reach Queue] timer expired before the [Message] arrived at the destination [Application Queue].

MQMSG_CLASS_NACK_Q_EXCEED_QUOTA: The message was generated to indicate that the [Message] was not inserted into the destination [Application Queue], because doing so would exceed the [Quota].

MQMSG_CLASS_NACK_ACCESS_DENIED: The message was generated to indicate that the [Message] was not inserted into the destination [Application Queue], because the user identified by [Message].[Sender Identifier] did not have sufficient rights to insert the [Message].

MQMSG_CLASS_NACK_HOP_COUNT_EXCEEDED: The message was generated to indicate that delivery of the [Message] was cancelled because it exceeded the maximum number of allowed routing hops. [<1>](#)

MQMSG_CLASS_NACK_BAD_SIGNATURE: The message was generated to indicate that the [Message] was not inserted into the destination [Application Queue], because the digital signature accompanying the [Message] was not successfully validated.

MQMSG_CLASS_NACK_BAD_ENCRYPTION: The message was generated to indicate that the [Message] was not inserted into the destination [Application Queue], because the [Message] could not be decrypted successfully.

MQMSG_CLASS_NACK_COULD_NOT_ENCRYPT: The message was generated to indicate that the [Message] was cancelled prior to delivery, because the [Message] could not be successfully encrypted.

MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_Q: The message was generated to indicate that the [Message] was not inserted into the destination [Application Queue], because the [Message] was sent as part of a transaction, but the destination [Application Queue].[IsTransactional] property equals False.

MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_MSG: The message was generated to indicate that the [Message] was not inserted into the destination [Application Queue], because the [Message] was not sent as part of a transaction, but the destination [Application Queue].[IsTransactional] property equals True.

MQMSG_CLASS_NACK_UNSUPPORTED_CRYPTO_PROVIDER: The message was generated to indicate that the [Message] was not inserted into the destination [Application Queue], because the destination [Queue Manager] does not support a cryptography library sufficient to decrypt the [Message] or validate its signature.

MQMSG_CLASS_NACK_SOURCE_COMPUTER_GUID_CHANGED: The message was generated to indicate that delivery of the [Message] was cancelled because the [GUID] property of the [Queue Manager] that originated the [Message] has changed. <2>

MQMSG_CLASS_NACK_Q_DELETED: The message was generated to indicate that the destination [Application Queue] was deleted before the [Message] could be received by a client of this protocol.

MQMSG_CLASS_NACK_Q_PURGED: The message was generated to indicate that the destination [Application Queue] was purged before the [Message] could be received by a client of this protocol.

MQMSG_CLASS_NACK_RECEIVE_TIMEOUT: The message was generated to indicate that the [Message].[Time To Be Received] timer expired before the [Message] could be received from the destination [Application Queue] by a client of this protocol.

MQMSG_CLASS_NACK_RECEIVE_TIMEOUT_AT_SENDER: The message was generated to indicate that the [Message].[Time To Be Received] timer expired before the [Message] could be inserted into the destination [Application Queue].

Used by:

- get [IMSMQMessage::Class](#)
- get [IMSMQMessage::MsgClass](#)
- put [IMSMQMessage::MsgClass](#)

2.2.2.10 MQMSGDELIVERY

The **MQMSGDELIVERY** enumeration defines values for the [Message].[Delivery Guarantee] property. The values of the enumeration indicate whether the [Message] will be recoverable in the event of a service interruption in the message queuing system.

```
typedef enum
{
    MQMSGDELIVERY_EXPRESS = 0x00000000,
    MQMSG_DELIVERY_RECOVERABLE = 0x00000001
} MQMSGDELIVERY;
```

MQMSGDELIVERY_EXPRESS: The [Message] will not be recovered in the event of a service interruption in the message queuing system. A client of this protocol selects this option if message throughput is preferred over the risk of message loss.

MQMSG_DELIVERY_RECOVERABLE: The [Message] SHOULD be recoverable in the event of most service interruptions in the message queuing system. A client of this protocol selects this

option to minimize the risk of message loss, even if the computer on which the [Message] resides crashes.

Used by:

- get [IMSMQMessage::Delivery](#)
- put [IMSMQMessage::Delivery](#)

2.2.2.11 MQMSGACKNOWLEDGEMENT

The **MQMSGACKNOWLEDGEMENT** enumeration defines flags for the [Message].[Acknowledgements Requested] property. The values of the enumeration indicate the categories of administrative acknowledgment messages that are generated in response to successful or unsuccessful delivery outcomes. Administrative acknowledgment messages are generated to indicate the delivery outcome of a message that was originated by a client of this protocol. The [MQMSGCLASS \(section 2.2.2.9\)](#) enumeration specifies the types of administrative acknowledgment messages and the specific conditions in which they are produced by the message queuing system.

```
typedef enum
{
    MQMSG_ACKNOWLEDGEMENT_NONE = 0x00000000,
    MQMSG_ACKNOWLEDGEMENT_POS_ARRIVAL = 0x00000001,
    MQMSG_ACKNOWLEDGEMENT_POS_RECEIVE = 0x00000002,
    MQMSG_ACKNOWLEDGEMENT_NEG_ARRIVAL = 0x00000004,
    MQMSG_ACKNOWLEDGEMENT_NEG_RECEIVE = 0x00000008,
    MQMSG_ACKNOWLEDGEMENT_NACK_REACH_QUEUE = 0x00000004,
    MQMSG_ACKNOWLEDGEMENT_FULL_REACH_QUEUE = 0x00000005,
    MQMSG_ACKNOWLEDGEMENT_NACK_RECEIVE = 0x0000000c,
    MQMSG_ACKNOWLEDGEMENT_FULL_RECEIVE = 0x0000000e
} MQMSGACKNOWLEDGEMENT;
```

MQMSG_ACKNOWLEDGEMENT_NONE: Administrative acknowledgment messages are not requested.

MQMSG_ACKNOWLEDGEMENT_POS_ARRIVAL: The message queuing system MUST generate an administrative acknowledgment message upon successful insertion into the [Messages Table] of the destination [Application Queue].

MQMSG_ACKNOWLEDGEMENT_POS_RECEIVE: An administrative acknowledgment message MUST be generated upon successful receipt of the [Message] from the destination [Application Queue].

MQMSG_ACKNOWLEDGEMENT_NEG_ARRIVAL: The message queuing system MUST generate an administrative acknowledgment message if the message cannot be successfully delivered to the destination [Application Queue].

MQMSG_ACKNOWLEDGEMENT_NEG_RECEIVE: The message queuing system MUST generate an administrative acknowledgment message if the message is not successfully received from the destination [Application Queue] by a client of this protocol.

MQMSG_ACKNOWLEDGEMENT_NACK_REACH_QUEUE: The message queuing system MUST generate an administrative acknowledgment message if the message cannot be successfully delivered to the destination [Application Queue].

MQMSG_ACKNOWLEDGEMENT_FULL_REACH_QUEUE: This value combines all the behaviors of MQMSG_ACKNOWLEDGEMENT_POS_ARRIVAL and MQMSG_ACKNOWLEDGEMENT_NEG_ARRIVAL.

MQMSG_ACKNOWLEDGEMENT_NACK_RECEIVE: This value combines all the behaviors of MQMSG_ACKNOWLEDGEMENT_NEG_ARRIVAL and MQMSG_ACKNOWLEDGEMENT_NEG_RECEIVE.

MQMSG_ACKNOWLEDGEMENT_FULL_RECEIVE: This value combines all the behaviors of MQMSG_ACKNOWLEDGEMENT_NEG_ARRIVAL, MQMSG_ACKNOWLEDGEMENT_NEG_RECEIVE, and MQMSG_ACKNOWLEDGEMENT_POS_RECEIVE.

Used by:

- get [IMSMQMessage::Ack](#)
- put [IMSMQMessage::Ack](#)

2.2.2.12 MQMSGJOURNAL

The **MQMSGJOURNAL** enumeration defines flags for the [Message].[Journaling Requested] property. The values of the enumeration indicate the source journaling mode for the [Message]. Source journaling is the process of retaining copies of messages that are sent. Two forms of source journaling are defined:

- **Positive source journaling:** The [Queue Manager] which sent the [Message] will retain a copy of the message only if it is successfully delivered.
- **Negative source journaling:** The [Queue Manager] which sent the [Message] will retain a copy of the message only if it is NOT successfully delivered. This behavior is also known as dead-lettering.

```
typedef enum
{
    MQMSG_JOURNAL_NONE = 0x00000000,
    MQMSG_DEADLETTER = 0x00000001,
    MQMSG_JOURNAL = 0x00000002
} MQMSGJOURNAL;
```

MQMSG_JOURNAL_NONE: Neither positive nor negative source journaling is requested for the [Message].

MQMSG_DEADLETTER: Negative source journaling is requested for the [Message].

If this value is specified for a [Message], the message queuing system MUST copy the message to the [System Deadletter Queue] if the [Message] is not successfully delivered to the destination [Application Queue]. If the [Quota] for the [System Deadletter Queue] is exceeded, the [Message] MUST be discarded.

MQMSG_JOURNAL: Positive source journaling is requested for the [Message].

If this value is specified for a [Message], the message queuing system MUST copy the message to the [System Journal Queue] if the [Message] is successfully delivered to the destination [Application Queue]. If the [Quota] for the [System Journal Queue] is exceeded, the [Message] MUST be discarded.

MQMSG_DEADLETTER and MQMSG_JOURNAL MAY be specified to enable both forms of source journaling.

Used by:

- get [IMSMQMessage::Journal](#)
- put [IMSMQMessage::Journal](#)

2.2.2.13 MQMSGTRACE

The **MQMSGTRACE** enumeration defines values that indicate whether the message tracing feature is enabled for a particular message. When message tracing is enabled for a [Message], the message transfer process will generate a report message for each hop along the route to the destination [Application Queue]. Report messages are administrative messages of type MQMSG_CLASS_REPORT, as specified by the [MQMSGCLASS \(section 2.2.2.9\)](#) enumeration.

```
typedef enum
{
    MQMSG_TRACE_NONE = 0x00000000,
    MQMSG_SEND_ROUTE_TO_REPORT_QUEUE = 0x00000001
} MQMSGTRACE;
```

MQMSG_TRACE_NONE: The message tracing feature of the message transfer process is disabled. This is the default value.

MQMSG_SEND_ROUTE_TO_REPORT_QUEUE: The message tracing feature of the message transfer process is enabled. [<3>](#)

Used by:

- get [IMSMQMessage::Trace](#)
- put [IMSMQMessage::Trace](#)

2.2.2.14 MQMSGSENDERIDTYPE

The **MQMSGSENDERIDTYPE** enumeration defines values for the [Message].[Sender Identifier Type] property. Specific values in this enumeration indicate the format of the [Sender Identifier] that is associated with a [Message].

```
typedef enum
{
    MQMSG_SENDERID_TYPE_NONE = 0x00000000,
    MQMSG_SENDERID_TYPE_SID = 0x00000001
} MQMSGSENDERIDTYPE;
```

MQMSG_SENDERID_TYPE_NONE: The identity of the sending user is not included in the [Message]. For the purposes of authorization, the sender identity for the [Message] is the **anonymous user**.

MQMSG_SENDERID_TYPE_SID: The identity of the sending user is indicated by the [Message].[Sender Identity] field that contains a **security identifier (SID)**.

Used by:

- get [IMSMQMessage::SenderIdType](#)
- put [IMSMQMessage::SenderIdType](#)

2.2.2.15 MQMSGPRIVLEVEL

The **MQMSGPRIVLEVEL** enumeration defines values for the [Message].[Privacy Level] property. Specific values in this enumeration indicate the manner in which a [Message] is to be encrypted when transmitted over the network by the message transfer process. A value is reserved to indicate that the [Message] is not to be encrypted.

```
typedef enum
{
    MQMSG_PRIV_LEVEL_NONE = 0x00000000,
    MQMSG_PRIV_LEVEL_BODY_BASE = 0x00000001,
    MQMSG_PRIV_LEVEL_BODY_ENHANCED = 0x00000003
} MQMSGPRIVLEVEL;
```

MQMSG_PRIV_LEVEL_NONE: The [Message] is not encrypted by the message transfer process.

MQMSG_PRIV_LEVEL_BODY_BASE: During the message transfer process, the [Body] of the [Message] MUST be protected from observation by using 40-bit encryption, as defined in [\[MS-MQOB\]](#) section 3.1.7.1.5.

MQMSG_PRIV_LEVEL_BODY_ENHANCED: During the message transfer process, the [Body] of the [Message] MUST be protected from observation by using 128-bit encryption, as defined in [\[MS-MQOB\]](#) section 3.1.7.1.5.

Used by:

- get [IMSMQMessage::PrivLevel](#)
- put [IMSMQMessage::PrivLevel](#)

2.2.2.16 MQMSGAUTHLEVEL

The **MQMSGAUTHLEVEL** enumeration defines values for the [Message].[Authentication Level] property. Specific values in this enumeration indicate the manner in which a [Message] is to be cryptographically signed when inserted in the [Outgoing Queue] by the [MSMQMessage4::Send](#) method. A value is reserved to indicate that the Send operation MUST NOT cryptographically sign the [Message].

```
typedef enum
{
    MQMSG_AUTH_LEVEL_NONE = 0x00000000,
    MQMSG_AUTH_LEVEL_ALWAYS = 0x00000001,
    MQMSG_AUTH_LEVEL_MSMQ10 = 0x00000002,
    MQMSG_AUTH_LEVEL_SIG10 = 0x00000002,
    MQMSG_AUTH_LEVEL_MSMQ20 = 0x00000004,
    MQMSG_AUTH_LEVEL_SIG20 = 0x00000004,
    MQMSG_AUTH_LEVEL_SIG30 = 0x00000008
}
```

```
} MQMSGAUTHLEVEL;
```

MQMSG_AUTH_LEVEL_NONE: The [Message] that was inserted into the [Outgoing Queue] by the Send operation is not digitally signed.

MQMSG_AUTH_LEVEL_ALWAYS: Prior to inserting the [Message] into the [Outgoing Queue], the Send operation MUST digitally sign the [Message]. For [Outgoing Queue]s that transfer to destination [Application Queue]s using the **SOAP Reliable Messaging Protocol (SRMP)** message transfer process, an **XML Digital Signature** MUST be created. For any other [Outgoing Queue], this value MUST be interpreted as equivalent to MQMSG_AUTH_LEVEL_SIG30. <4>

MQMSG_AUTH_LEVEL_MSMQ10: Prior to inserting the [Message] into the [Outgoing Queue], the Send operation MUST digitally sign the [Message] according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x1.

MQMSG_AUTH_LEVEL_SIG10: Prior to inserting the [Message] into the [Outgoing Queue], the Send operation MUST digitally sign the [Message] according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x1.

MQMSG_AUTH_LEVEL_MSMQ20: Prior to inserting the [Message] into the [Outgoing Queue], the Send operation MUST digitally sign the [Message] according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x3.

MQMSG_AUTH_LEVEL_SIG20: Prior to inserting the [Message] into the [Outgoing Queue], the Send operation MUST digitally sign the [Message] according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x3.

MQMSG_AUTH_LEVEL_SIG30: Prior to inserting the [Message] into the [Outgoing Queue], the Send operation MUST digitally sign the [Message] according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x5.

Used by:

- get [IMSMQMessage::AuthLevel](#)
- put [IMSMQMessage::AuthLevel](#)

2.2.2.17 MQMSGAUTHENTICATION

The **MQMSGAUTHENTICATION** enumeration defines values for the [Message].[Authentication Level] property. Specific values in this enumeration indicate the manner in which a [Message] was cryptographically signed. A value is reserved to indicate that the [Message] was not signed.

```
typedef enum
{
    MQMSG_AUTHENTICATION_NOT_REQUESTED = 0x00000000,
    MQMSG_AUTHENTICATION_REQUESTED = 0x00000001,
    MQMSG_AUTHENTICATED_SIG10 = 0x00000001,
    MQMSG_AUTHENTICATION_REQUESTED_EX = 0x00000003,
    MQMSG_AUTHENTICATED_SIG20 = 0x00000003,
    MQMSG_AUTHENTICATED_SIG30 = 0x00000005,
    MQMSG_AUTHENTICATED_SIGXML = 0x00000009
} MQMSGAUTHENTICATION;
```


MQMSG_AUTHENTICATION_NOT_REQUESTED: The [Message] was not signed.

MQMSG_AUTHENTICATION_REQUESTED: The [Message] was signed according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x1.

MQMSG_AUTHENTICATED_SIG10: The [Message] was signed according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x1.

MQMSG_AUTHENTICATION_REQUESTED_EX: The [Message] was signed according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x3.

MQMSG_AUTHENTICATED_SIG20: The [Message] was signed according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x3.

MQMSG_AUTHENTICATED_SIG30: The [Message] was signed according to the algorithm described in [\[MS-MQOB\]](#) section 2.2.8.6, where the SecurityHeader.Flags.AS value is 0x5.

MQMSG_AUTHENTICATED_SIGXML: The [Message] was signed using an XML Digital Signature.

Used by:

- get [IMSMQMessage::ReceivedAuthenticationLevel](#)

2.2.2.18 MQCALG

The **MQCALG** enumeration defines numeric values that represent specific cryptographic encryption and hash algorithms.

```
typedef enum
{
    MQMSG_CALG_MD2 = 0x00008001,
    MQMSG_CALG_MD4 = 0x00008002,
    MQMSG_CALG_MD5 = 0x00008003,
    MQMSG_CALG_SHA = 0x00008004,
    MQMSG_CALG_SHA1 = 0x00008004,
    MQMSG_CALG_MAC = 0x00008005,
    MQMSG_CALG_RSA_SIGN = 0x00002400,
    MQMSG_CALG_DSS_SIGN = 0x00002200,
    MQMSG_CALG_RSA_KEYX = 0x0000a400,
    MQMSG_CALG_DES = 0x00006601,
    MQMSG_CALG_RC2 = 0x00006602,
    MQMSG_CALG_RC4 = 0x00006801,
    MQMSG_CALG_SEAL = 0x00006802
} MQCALG;
```

MQMSG_CALG_MD2: The MD2 Message-Digest Algorithm, as defined in [\[RFC1319\]](#). This value is not supported. [<5>](#)

MQMSG_CALG_MD4: The MD4 Message-Digest Algorithm, as defined in [\[RFC1320\]](#). This value is not supported. [<6>](#)

MQMSG_CALG_MD5: The MD5 Message-Digest Algorithm, as defined in [\[RFC1321\]](#). This value is not supported. [<7>](#)

MQMSG_CALG_SHA: The US Secure Hash Algorithm 1 (SHA1), as defined in [\[RFC3174\]](#).

MQMSG_CALG_SHA1: The US Secure Hash Algorithm 1 (SHA1), as defined in [\[RFC3174\]](#).

MQMSG_CALG_MAC: Not supported. [<8>](#)

MQMSG_CALG_RSA_SIGN: The RSA signature algorithm, as defined in [\[PKCS1\]](#).

MQMSG_CALG_DSS_SIGN: The Digital Signature Standard (DSS), as defined in [\[FIPS186\]](#).

MQMSG_CALG_RSA_KEYX: The RSA key exchange algorithm, as defined in [\[PKCS1\]](#).

MQMSG_CALG_DES: The Data Encryption Standard (DES), as defined in [\[FIPS46-3\]](#).

MQMSG_CALG_RC2: The RC2 Encryption Algorithm, as defined in [\[RFC2268\]](#).

MQMSG_CALG_RC4: The RC4 stream cipher, as defined in [\[RFC4757\]](#).

MQMSG_CALG_SEAL: The Software-Optimized Encryption Algorithm, as defined in [\[SEAL-SPRINGER\]](#).

Used by

- get [IMSMQMessage::HashAlgorithm](#)
- put [IMSMQMessage::HashAlgorithm](#)
- get [IMSMQMessage::EncryptAlgorithm](#)
- put [IMSMQMessage::EncryptAlgorithm](#)

2.2.2.19 QUEUE_STATE

The **QUEUE_STATE** enumeration defines values that indicate the status of an [Outgoing Queue]. The state of an [Outgoing Queue] is determined and managed by the message transfer process.

```
typedef enum
{
    MQ_QUEUE_STATE_LOCAL_CONNECTION = 0x00000000,
    MQ_QUEUE_STATE_DISCONNECTED = 0x00000001,
    MQ_QUEUE_STATE_WAITING = 0x00000002,
    MQ_QUEUE_STATE_NEEDVALIDATE = 0x00000003,
    MQ_QUEUE_STATE_ONHOLD = 0x00000004,
    MQ_QUEUE_STATE_NONACTIVE = 0x00000005,
    MQ_QUEUE_STATE_CONNECTED = 0x00000006,
    MQ_QUEUE_STATE_DISCONNECTING = 0x00000007,
    MQ_QUEUE_STATE_LOCKED = 0x00000008
} QUEUE_STATE;
```

MQ_QUEUE_STATE_LOCAL_CONNECTION: The [Outgoing Queue] exists to contain messages pending transfer to a [Queue] that is owned by the same [Queue Manager].

MQ_QUEUE_STATE_DISCONNECTED: The message transfer process has no connected network sessions for transferring messages. [Message]s in the [Outgoing Queue] are not being transferred to their destination [Application Queue](s). The message transfer process MAY attempt to connect and transfer messages at a later time.

MQ_QUEUE_STATE_WAITING: The message transfer process is attempting to connect and transfer messages to a destination [Application Queue], but a response has not been received. [Message]s in the [Outgoing Queue] are not yet being transferred to their destination [Application Queue](s).

MQ_QUEUE_STATE_NEEDVALIDATE: The message transfer process is unable to retrieve properties of the destination [Application Queue] prior to initiating message transfer. This state is typically achieved when the [Queue Manager] that owns the [Outgoing Queue] has [IsConnected] equal to False.

[Message]s in the [Outgoing Queue] are not being transferred to their destination [Application Queue](s).

MQ_QUEUE_STATE_ONHOLD: A client of this protocol has paused the [Outgoing Queue]. [Message]s in the [Outgoing Queue] are not being transferred to their destination [Application Queue](s).

MQ_QUEUE_STATE_NONACTIVE: The message transfer process has no connected network sessions for transferring messages. [Message]s in the [Outgoing Queue] are not being transferred to their destination [Application Queue](s). The message transfer process will not attempt to connect and transfer messages.

MQ_QUEUE_STATE_CONNECTED: The message transfer process is currently transferring messages to the destination [Application Queue].

MQ_QUEUE_STATE_DISCONNECTING: The message transfer process is currently disconnecting from the destination [Application Queue]. The message transfer process is not transmitting messages in this state.

MQ_QUEUE_STATE_LOCKED: The system configuration prevents the message transfer process from transferring messages to the destination [Application Queue].

Used by:

- get [IMSMQOutgoingQueueManagement::State](#)

2.2.2.20 RELOPS

The **RELOPS** enumeration defines values for the [MSMQQuery::LookupQueue](#) and [MSMQQuery::LookupQueue_v2](#) methods. The values in this enumeration represent comparison operators used in conjunction with the query parameters passed to **MSMQQuery::LookupQueue**.

```
typedef enum
{
    REL_NOP = 0x00000000,
    REL_EQ = 0x00000001,
    REL_NEQ = 0x00000002,
    REL_LT = 0x00000003,
    REL_GT = 0x00000004,
    REL_LE = 0x00000005,
    REL_GE = 0x00000006
} RELOPS;
```

REL_NOP: The associated query parameter MUST be ignored.

REL_EQ: The query MUST match [Directory Queue]s for which the specified query parameter equals the corresponding value in the [Directory Queue].

REL_NEQ: The query MUST match [Directory Queue]s for which the specified query parameter is not equal to the corresponding value in the [Directory Queue].

REL_LT: The query MUST match [Directory Queue]s for which the specified query parameter is less than the corresponding value in the [Directory Queue].

REL_GT: The query MUST match [Directory Queue]s for which the specified query parameter is greater than the corresponding value in the [Directory Queue].

REL_LE: The query MUST match [Directory Queue]s for which the specified query parameter is less than, or equal to, the corresponding value in the [Directory Queue].

REL_GE: The query MUST match [Directory Queue]s for which the specified query parameter is greater than, or equal to, the corresponding value in the [Directory Queue].

Used by:

- **IMSMQQuery::LookupQueue**
- **IMSMQQuery::LookupQueue_v2**

2.2.2.21 XACTTC

The **XACTTC** enumeration defines the commit behavior of a transaction. The values in this enumeration indicate the synchronous, asynchronous, or two-phased behavior of the transaction.

```
typedef enum
{
    XACTTC_NONE = 0x00000000,
    XACTTC_SYNC_PHASEONE = 0x00000001,
    XACTTC_SYNC_PHASETWO = 0x00000002,
    XACTTC_SYNC = 0x00000002,
    XACTTC_ASYNC_PHASEONE = 0x00000004,
    XACTTC_ASYNC = 0x00000004
} XACTTC;
```

XACTTC_NONE: The default commit behavior of the transaction coordinator is used.

XACTTC_SYNC_PHASEONE: The commit method returns after **phase one** of the **two-phase commit** has been completed.

XACTTC_SYNC_PHASETWO: The commit method returns after **phase two** of the two-phase commit has completed.

XACTTC_SYNC: The commit method returns after phase two of the two-phase commit has completed.

XACTTC_ASYNC_PHASEONE: The **commit request** is performed asynchronously.

XACTTC_ASYNC: The commit request is performed asynchronously.

Used by:

- [**ITransaction::Commit**](#)

- [IMSMQTransaction::Commit](#)

2.2.3 Data Collections

The following sections describe name-value pair sets returned by the [MSMQQueueManagement::EodGetReceiveInfo \(section 3.4.4.1.3\)](#) and [MSMQOutgoingQueueManagement::EodGetSendInfo \(section 3.5.4.1.3\)](#) methods.

2.2.3.1 EODTargetInfo

The EODTargetInfo data collection defines the elements of the [EODTargetInfos] array property of an [Application Queue]. Each EODTargetInfo is an [MSMQCollection \(section 3.15\)](#) object that contains the name-value pairs described below.

Property name	Property value data type	Property description
QueueFormatName	VT_BSTR	The [Destination Format Name] of the [Outgoing Queue] that transfers messages to the [Application Queue].
SenderID	VT_BSTR	The [GUID] property of the [Queue Manager] that is transferring messages from one of its [Outgoing Queue]s to the [Application Queue].
SeqID	VT_UI8	The 64-bit unsigned integer value that identifies the last transactional message sequence that was sent to the [Application Queue]. This property corresponds to the IncomingTxSequenceID property described in [MS-MQOB] section 3.1.1.3 .
SeqNo	VT_I4	The 32-bit value that identifies the sequence number of the last transactional message that was received by the [Application Queue] in a particular transfer session. This property corresponds to the IncomingTxSequenceNumber property described in [MS-MQOB] section 3.1.1.3 .
LastAccessTime	VT_I4	The time of the last activity on the [Application Queue] when the message was either accepted or rejected, formatted as a time_t.
RejectCount	VT_I4	The number of times the message was rejected by the reception end of the message transfer process, before being accepted into the [Application Queue].

Used by:

- [IMSMQQueueManagement::EodGetReceiveInfo](#)

2.2.3.2 EODSourceInfo

The EODSourceInfo data collection defines the information contained in the [EOD Source] property of an [Outgoing Queue]. The EODSourceInfo is an [MSMQCollection \(section 3.15\)](#) that contains the name-value pairs described below.

Property name	Property value data type	Property description
EodLastAckCount	VT_UI4	Specifies the number of times that the last order acknowledgment was received by the [Queue Manager] that owns the [Outgoing Queue] that is transferring messages to the destination [Application Queue]. For more details on the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2.
EodLastAckTime	VT_I4	Specifies the time when the last order acknowledgment was received by the [Queue Manager] that owns the [Outgoing Queue] that is transferring messages to the destination [Application Queue]. For more details on the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2.
EodNoAckCount	VT_UI4	Specifies the number of messages sent from the [Outgoing Queue] for which an order acknowledgment has not been received. For more details on the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2.
EodNoReadCount	VT_UI4	Specifies the number of messages sent from the [Outgoing Queue] for which an order acknowledgment has been received, but a final acknowledgment has not been received. For the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2; for the final acknowledgment, refer to [MS-MQOB] section 3.1.7.2.2.
EodResendCount	VT_UI4	Specifies the number of times that the last message was transferred from the [Outgoing Queue].
EodResendInterval	VT_UI4	Specifies the time interval for resending the messages in the [Outgoing Queue] for which no order acknowledgment has been received. For more details on the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2.
EodResendTime	VT_I4	A time_t value indicating the time when the messages in the [Outgoing Queue] for which no order acknowledgment has been received will be retransmitted by the message transfer process. For more details on the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2.
EodFirstNonAck	SequenceInfoCollection	A SequenceInfoCollection (section 2.2.3.3) data collection containing sequence information about the first transactional message that was transferred from the [Outgoing Queue] for which an order acknowledgment has not been received. For more details on the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2.
EodLastAck	SequenceInfoCollection	A SequenceInfoCollection (section 2.2.3.3) data collection containing sequence information about the last transactional message that was transferred from the [Outgoing Queue] for which an order acknowledgment has been received. For more details on the order acknowledgment, see [MS-MQOB] section 3.1.1.4.2.

Property name	Property value data type	Property description
EodLastNonAck	SequenceInfoCollection	A SequenceInfoCollection (section 2.2.3.3) data collection containing sequence information about the last transactional message that was transferred from the [Outgoing Queue] for which an order acknowledgment has not been received. For more details on the order acknowledgment, see [MS-MQQB] section 3.1.1.4.2.
EodNextSeq	SequenceInfoCollection	A SequenceInfoCollection (section 2.2.3.3) data collection containing sequence information about the next message to be transferred from the [Outgoing Queue].

Used by:

- [IMSMQOutgoingQueueManagement::EodGetSendInfo](#)

2.2.3.3 SequenceInfoCollection

The SequenceInfoCollection data collection defines transactional message transfer sequence information contained in the EodFirstNonAck, EodLastAck, EodLastNonAck and EodNextSeq name-value pairs contained in the [MSMQCollection \(section 3.15\)](#) object that represents the [EOD Target] property of an [Outgoing Queue]. Transactional message transfer sequences are defined in [\[MS-MQQB\]](#) section 3.1.1.3. The SequenceInfoCollection is an MSMQCollection object that contains the name-value pairs described below.

Property name	Property value data type	Property description
SeqID	VT_UI8	The 64-bit value that identifies the current transactional message sequence being transferred from the [Outgoing Queue]. This property corresponds to the TxSequenceID property, as described in [MS-MQQB] section 3.1.1.3.
SeqNo	VT_I4	The 32-bit value that identifies the sequence number of the last transactional message that was transferred by the [Outgoing Queue] in a particular transfer session. This property corresponds to the TxSequenceNumber property, as described in [MS-MQQB] section 3.1.1.3.
PrevNo	VT_I4	The 32-bit value that identifies the sequence number of the previous transactional message that was transferred by the [Outgoing Queue] in a particular transfer session.

Used by:

- [IMSMQOutgoingQueueManagement::EodGetSendInfo](#)

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 Common Implementation Details

This section describes common implementation details applicable to the MSMQ: ActiveX Client Protocol, including the system data model expressed in conceptual terms (the abstract data model). The system data model represents an abstract message queuing system with which clients of this protocol interact.

All object classes defined by the MSMQ: ActiveX Client Protocol MUST inherit from, and implement, the standard IUnknown and IDispatch interfaces, as defined in [\[MS-DCOM\]](#) section **3.2.1.5.8**, and [\[MS-OAUT\]](#) section **3.1**, respectively. Additional information can be found in [Box98]. For each object class defined below, the methods of IUnknown represent method **opnums** 0–2. The methods of IDispatch are opnums 3–6; hence the first defined opnum for each object class defined in sections [3.2](#) to [3.17](#) below is 7.

Section [2.2](#) describes the particular data types, such as enumerations and data collections, that are referred to by elements of the system data model. Sections [3.2](#) - [3.17](#) describe the client protocol and respective server behavior in terms of manipulating the system data. These sections use a property accessor method name disambiguation convention where the prefix "put_" denotes a "propput" invocation type, "get_" denotes a "propget" invocation type, and "putref_" denotes a "propputref" invocation. For more information on OLE Automation invocation types, refer to [MS-OAUT] sections [2.2.11](#) and [2.2.45.5.1](#)

3.1.1 Abstract Data Model

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

The concepts defined below form the basis on which the behaviors mandated by this protocol are specified.

3.1.1.1 Directory

A [Directory] is a read-write database that contains public information regarding [Directory Queue Manager]s, [Directory Queue]s, and [Directory User]s. A [Queue Manager] MAY be affiliated with a single [Directory] at a time, but never multiple [Directory]s simultaneously. This protocol does not mandate the means by which the directory affiliation is established.

A [Directory] MUST contain the following elements: [<9>](#)

- [Directory Queue Managers Table]: An instance of a [Directory Queue Managers Table] that contains the list of [Directory Queue Manager]s exposed by the [Directory].
- [Directory Users Table] : An instance of a [Directory Users Table] that contains the list of [Directory User]s exposed by the [Directory].

3.1.1.2 Directory Users Table

A [Directory Users Table] MUST contain an associative list of zero or more [Directory User]s indexed by [Directory User].[Name].

3.1.1.3 Directory User

A [Directory User] associates cryptographic **certificate** information with a specific user of the message queuing system. Implementations of the message transfer process MAY verify the signature of a [Message] that was signed by obtaining a cryptographic [Certificate] from the [Certificates Table] by locating the [Directory User] for which the [Security Identifier] equals the [Sender Identifier] of the [Message].

A [Directory User] MUST contain the following elements:

- [Security Identifier]: A SID, as specified in [\[MS-DTYP\]](#) section **2.4.2**.
- [Certificates Table]: An instance of a [Certificates Table] that contains the list of each [Certificate] that is registered for the [Directory User].

3.1.1.4 Certificates Table

A [Certificates Table] MUST contain an associative list of zero or more [Certificate]s indexed by [Certificate].[Digest].

3.1.1.5 Certificate

A [Certificate] contains a standard X.509 [\[X509\]](#) certificate, and a digest that uniquely identifies the [Certificate].

A [Certificate] MUST contain the following elements:

- [X509]: An X.509–encoded certificate, as defined in [\[RFC3280\]](#).
- [Digest]: An MD5 message digest computed from the [Certificate], as defined in [\[RFC1321\]](#).

The following types are defined as specializations of [Certificate]:

- [Internal Certificate]
- [External Certificate]

3.1.1.6 Internal Certificate

An [Internal Certificate] is a system-generated [Certificate] that is stored in an **internal certificate store** on the server. There MUST be at most one [Internal Certificate] in the internal certificate store on the server computer. An [Internal Certificate] MUST be registered in the [Directory] for a [Directory User], implicitly by the system, [<10>](#) or explicitly by the client.

3.1.1.7 External Certificate

An [External Certificate] is a client-specified [Certificate] that MUST be registered in the **personal certificate store** on the server. An [External Certificate] MUST be registered in the [Directory] for a [Directory User] explicitly by the client.

3.1.1.8 Directory Queue Managers Table

A [Directory Queue Managers Table] MUST contain an associative list of zero or more [Directory Queue Manager]s indexed by [Directory Queue Manager].[Computer Name].

3.1.1.9 Directory Queue Manager

A [Directory Queue Manager] exposes information about a [Queue Manager]. A [Directory Queue Manager] exists for each [Queue Manager] that is affiliated with the [Directory].

Each [Directory Queue Manager] MUST contain the following elements:

- [Computer Name]: A string that MUST uniquely identify the [Directory Queue Manager] within the [Directory Queue Managers Table]. [Computer Name] MUST be equal to the [Queue Manager].[Computer Name] for the [Queue Manager] that this [Directory Queue Manager] exposes.
- [GUID]: A GUID that MUST uniquely identify the [Directory Queue Manager] within the [Directory Queue Managers Table]. As with [Computer Name], [GUID] MUST be equal to [Queue Manager].[GUID] for the [Queue Manager] that is exposed by this [Directory Queue Manager].
- [Directory Queues Table]: An instance of a [Directory Queues Table] that contains the list of [Directory Queue]s exposed by the [Directory Queue Manager].

3.1.1.10 Directory Queues Table

A [Directory Queues Table] MUST contain an associative list of zero or more [Directory Queue]s indexed by [Directory Queue].[Name].

3.1.1.11 Directory Queue

A [Directory Queue] exposes information about a [Public Queue] that is owned by the [Queue Manager] that corresponds to the [Directory Queue Manager] in which the [Directory Queue] is contained.

A [Directory Queue] MUST contain the following elements:

- [Name]: A string that MUST uniquely identify the [Directory Queue] within the scope of the [Directory Queues Table] of the [Directory Queue Manager]. [Name] corresponds with the [Name] property of the exposed [Public Queue].
- [Path]: A directory-specific address string that MUST uniquely identify the [Directory Queue] within the scope of the entire [Directory]. [<11>](#)

Additionally, a [Directory Queue] MUST expose the values of the following properties of the corresponding [Public Queue]:

- [Identifier GUID]
- [Service Type GUID]
- [Label]
- [IsTransactional]
- [IsWorldReadable]

- [IsTargetJournalingEnabled]
- [Quota]
- [Journal Queue].[Quota]
- [Creation Time]
- [Modification Time]
- [Authentication Required]
- [Multicast Address]
- [Privacy Level]

3.1.1.12 Queue Manager

A [Queue Manager] represents a computer acting in the message queuing **independent client** role. At most, one [Queue Manager] exists on any named computer. A [Queue Manager] MAY be affiliated with a [Directory], in which case a [Directory Queue Manager] MUST exist in the affiliated [Directory]. Conversely, a [Queue Manager] MAY exist independently of a [Directory]—an operating mode that is defined by [\[MS-MQMA\]](#) section 2.2.1.1.2.

A specific instance of [Queue Manager], termed [Local Queue Manager], refers to the single [Queue Manager] affiliated with the computer at the server end of this protocol.

A [Queue Manager] MUST contain the following elements:

- [Computer Name]: A string containing the name of the computer where the [Queue Manager] is present.
- [GUID]: A GUID that MUST uniquely identify the [Queue Manager]. This value MUST NOT change.
- [IsDirectoryEnabled]: A Boolean flag that, when True, indicates that the [Queue Manager] is affiliated with a [Directory]. When False, this flag indicates that the [Queue Manager] operates independently of a [Directory]. This protocol does not mandate the process through which a [Queue Manager] becomes affiliated with a [Directory].
- [IsConnected]: A Boolean flag that, when true, indicates that the [Queue Manager] is listening for incoming message transfer sessions from other [Queue Manager]s. For [Queue Manager]s affiliated with a [Directory], this value also indicates that the [Queue Manager] is able to successfully communicate with the [Directory].
- [Queues Table]: A list of [Queue]s that are owned by the [Queue Manager].
- [Open Queues Table]: A list of [Open Queue]s that are approved by the [Queue Manager].
- [Enlisted Transactions Table]: A list of [Enlisted Transaction]s that are recognized by the [Queue Manager].
- [System Deadletter Queue]: A [System Queue] that contains [Message]s that were not delivered.
- [System Journal Queue]: A [System Queue] that contains copies of successfully delivered [Message]s.

3.1.1.13 Local Queue Manager

The MSMQ: ActiveX Client Protocol mandates that exactly one [Queue Manager] MUST be affiliated with the server computer. Throughout this specification, the term [Local Queue Manager] is taken to refer to the [Queue Manager] that is affiliated with the server computer.

3.1.1.14 Queues Table

A [Queues Table] MUST contain an associative list of zero or more [Queue]s indexed by [Queue].[Name].

3.1.1.15 Queue

A [Queue] is owned by at most one [Queue Manager], and contains a [Messages Table].

[Queue] is an abstract base element; therefore only particular specializations of [Queue] are instantiatable. The following elements are defined as specializations of [Queue]:

- [Application Queue] (also an abstract base element)
 - [Public Queue]
 - [Private Queue]
- [Journal Queue]
- [System Queue]
- [Outgoing Queue]

A [Queue] MUST contain the following elements:

- [Name]: A string that MUST uniquely identify the [Queue] within the scope of the [Queues Table] of the [Queue Manager].
- [Messages Table]: An instance of a [Messages Table] that contains the set of [Message]s in the [Queue].
- [Creation Time]: A date/time value that indicates when the [Queue] was created.
- [IsTransactional]: A Boolean flag that, when True, indicates that the [Queue] is a transactional queue. This value MUST be determined when the [Queue] is created, and MUST NOT be modified.
- [IsActive]: A read-only Boolean flag that, when retrieved, MUST be calculated using the following rule. [IsActive] is True if any of the following conditions are true:
 1. [Messages Table] contains one or more [Message]s.
 2. One or more [Open Queue]s reference the [Queue].

3.1.1.16 Application Queue

An [Application Queue] is a specialized [Queue] that clients can receive messages from and send messages to. Further specializations of [Application Queue] are created directly by clients of this protocol; as opposed to other kinds of [Queue]s created indirectly as a function of the message queuing system.

[Application Queue] is an abstract base element; therefore only particular specializations of [Application Queue] are instantiatable. The following types are defined as specializations of [Application Queue]:

- [Public Queue]
- [Private Queue]

In addition to the elements defined for [Queue], an [Application Queue] MUST contain the following elements:

- [Journal Queue]: A [Journal Queue] records messages that are received from the [Application Queue] while [IsTargetJournalingEnabled] is True.
- [Owner]: A security identifier (SID) that indicates the user account that is permitted to modify properties of the [Application Queue].
- [Identifier GUID]: A GUID that uniquely identifies the [Application Queue]. This value is assigned when the [Application Queue] is created, and MUST NOT be modified.
- [Service Type GUID]: A GUID that clients of this protocol MAY use to classify the [Application Queue] in an application-specific manner. The value is not required to be unique at any scope; therefore several [Application Queue]s MAY have identical [Service Type GUID] values.
- [Label]: A text string that MAY contain a client-defined description for the [Application Queue]. [Label] MUST NOT exceed 124 characters in length.
- [IsWorldReadable]: A Boolean flag that, when True, indicates that the [Application Queue] is accessible to any user. When False, this flag indicates that the [Application Queue], and its associated [Journal Queue], MUST only be accessible to **administrators** and the [Owner].
- [IsTargetJournalingEnabled]: A Boolean flag that, when True, indicates that messages received from the [Messages Table] are recorded in the [Journal Queue]. [Messages Table].
- [Quota]: A value that indicates the maximum size of the [Messages Table], expressed in kilobytes.
- [Modification Time]: A date/time value that indicates when any element of the [Application Queue] was most recently modified.
- [Authentication Required]: A Boolean flag that, when True, indicates that the [Application Queue] accepts only [Message]s for which the [Authentication Level] indicates that the [Message] was signed. When False, the [Application Queue] does not mandate that messages must be signed.
- [Multicast Address]: A string containing an IP multicast address on which the respective [Queue Manager] will receive [Message]s for this [Application Queue]. By default, this value is blank (""), indicating that the [Application Queue] is not associated with an IP multicast address. For the purposes of this protocol specification, this is an opaque read/write value. The behavior affected by this value is particular to the implementation of the message queuing system. [<12>](#)
- [Privacy Level]: An enumeration that indicates whether the [Application Queue] restricts [Message]s according to their [Privacy Level] value. The enumeration values are defined in section [2.2.2.7](#).
- [EODTargetInfos]: An array of zero or more [EODTargetInfo \(section 2.2.3.1\)](#) structures, containing statistics about each of the individual message transfer processes that are delivering messages to the [Application Queue].

3.1.1.17 Public Queue

A [Public Queue] is a specialized [Application Queue] that advertises its presence via the existence of a corresponding [Directory Queue] in the [Directory]. Since a [Public Queue] is associated with a [Directory Queue], a [Queue Manager] that lacks [Directory] affiliation therefore cannot own a [Public Queue].

In addition to the elements defined for [Queue] and [Application Queue], a [Public Queue] MUST contain the following elements:

- [Base Priority]: A numeric value between -32768 and 32767 (the default is zero). Implementations of the message transfer process MAY interpret [Base Priority] relative to other [Application Queue]s to prioritize message transfer work. A greater value indicates higher priority. [Private Queue]s MUST be considered to have a [Base Priority] value of zero (0).

3.1.1.18 Private Queue

A [Private Queue] is a specialized [Application Queue] that does not advertise its presence beyond its [Queue Manager].

3.1.1.19 Outgoing Queue

An [Outgoing Queue] is a specialized [Queue] that contains messages that are awaiting transfer to one or more [Application Queue]s. Messages are never explicitly sent to an [Outgoing Queue]; rather, messages are implicitly placed in an [Outgoing Queue] by the message queueing system, pending transfer to a destination [Application Queue].

In addition to the elements defined for [Queue], an [Outgoing Queue] MUST contain the following elements:

- [Destination Format Name]: A format name that identifies zero or more destination [Application Queue]s for [Message]s in the [Outgoing Queue]. The value of the [Queue].[Name] property MUST be equal to the value of this property.
- [Outgoing Queue State]: An enumerated value that indicates the status of the [Outgoing Queue] for the purposes of the message transfer process. The defined values are contained in the [QUEUE_STATE \(section 2.2.2.19\)](#) enumeration.
- [Next Hops]: An array of strings that contains one or more possible addresses for routing messages to the destination [Application Queue]. The address string formats are defined in [\[MS-MQMQ\]](#) section 2.3.12.12.
- [EODSourceInfo]: An [EODSourceInfo \(section 2.2.3.2\)](#) structure that contains statistics about the message transfer process.

3.1.1.20 System Queue

A [System Queue] is a [Queue] that is created by the [Queue Manager] to record [Message]s transferred from an [Outgoing Queue], and [Message]s unable to be transferred. Both [Queue Manager].[System Deadletter Queue] and [Queue Manager].[System Journal Queue] are [System Queue]s.

In addition to the elements defined for [Queue], a [System Queue] MUST contain the following elements:

- [Quota]: A value that indicates the maximum size of the [Messages Table], expressed in kilobytes.

3.1.1.21 Journal Queue

A [Journal Queue] contains messages that were removed from an [Application Queue]. Exactly one [Journal Queue] exists for each [Application Queue].

In addition to the elements defined for [Queue], a [Journal Queue] MUST contain the following elements:

- [Quota]: A value that indicates the maximum size of the [Messages Table], expressed in kilobytes.

3.1.1.22 Messages Table

A [Messages Table] MUST contain an ordered, associative list of zero or more [Message]s indexed and sorted in ascending order by [Message].[Lookup Identifier]. A [Messages Table] features a head and a tail, and is bidirectionally iterable.

A [Messages Table] MUST contain the following elements:

- [Message Count]: A read-only numeric value calculated at the time of retrieval. It MUST contain the number of messages in the [Messages Table].
- [Total Bytes]: A read-only numeric value calculated at the time of retrieval. It MUST contain the amount of storage space consumed by all the messages in the [Messages Table], as would be charged against the queue quota, expressed in bytes.

3.1.1.23 Message

A [Message] represents a unit of data transfer between clients of this protocol.

A [Message] MUST contain the following elements:

- [Lookup Identifier]: An unsigned integer that both uniquely identifies, and indicates the relative position of, the [Message] within the scope of its [Messages Table]. The value of the [Lookup Identifier] MUST be determined based on ascending order of [Message].[Arrival Time] within [Message].[Priority]. The value of the [Lookup Identifier] for a [Message] at the head of the [Messages Table] MUST be the lowest in the [Messages Table]; conversely, the [Message] at the tail MUST have the greatest [Lookup Identifier] value. The [Lookup Identifier] MUST be assigned by the [Queue Manager] when the [Message] is added to the [Messages Table].
- [Message Identifier]: An OBJECTID, as specified in [\[MS-MQMQ\]](#) section 2.2.8, that uniquely identifies the [Message] in the scope of the entire message queuing system. This value is assigned by the [MSMQMessage4::Send](#) method.
- [IsLocked]: A Boolean flag that, when True, indicates that the Message is in use by a [Transactional Operation], and therefore MUST NOT be visible to clients unless [AllowPeekWhenLocked] is True.
- [AllowPeekWhenLocked]: A Boolean flag that, when True, indicates that the [message] MUST be visible through peek operations, even if [IsLocked] is True.

- [Arrival Time]: A Coordinated Universal Time (UTC) date/time value that indicates when the [Message] was added to the [Messages Table]. This value is assigned when the [Message] is added to the [Messages Table].
- [Class]: An enumerated value that indicates the message classification. The defined values are contained in the [MQMSGCLASS \(section 2.2.2.9\)](#) enumeration.
- [Privacy Level]: An enumerated value that indicates the manner in which the message transfer process must encrypt the [Body] during message transfer process. The defined values are contained in the [MQMSGPRIVLEVEL \(section 2.2.2.15\)](#) enumeration.
- [Authentication Level]: An enumerated value that indicates the manner in which the message was cryptographically signed. The defined values are contained in the [MQMSGAUTHENTICATION \(section 2.2.2.17\)](#) enumeration.
- [Delivery Guarantee]: An enumerated value that indicates whether the [Message] will be recoverable in the event of a service interruption in the message queuing system. The defined values are contained in the [MQMSGDELIVERY \(section 2.2.2.10\)](#) enumeration.
- [Tracing Requested]: An enumerated value that indicates whether the route tracing feature of the message transfer process is enabled for the [Message]. The defined values are contained in the [MQMSGTRACE \(section 2.2.2.13\)](#) enumeration.
- [Priority]: A numeric value, from zero (0) to seven (7), that is used as an input to the determination of [Lookup Identifier]. Zero (0) is the highest priority, and seven (7) is the lowest; therefore, high-priority [Message]s are encountered before low-priority messages when forward-iterating the messages in a queue.
- [Journaling Requested]: A numeric value that contains bitflags that indicate the source journaling mode for the [Message]. The defined values are contained in the [MQMSGJOURNAL \(section 2.2.2.12\)](#) enumeration.
- [Response Queue Format Name]: A client-provided format name string that identifies a **response queue** [Application Queue]. Clients of this protocol MAY use this field to specify an application-specific "reply-to address" for the [Message]. The server MUST not interpret or modify this field.
- [AppSpecific]: A client-provided numeric value. Clients of this protocol MAY use this field for any application-specific purpose. The server MUST not interpret or modify this field.
- [Source Machine GUID]: A GUID that contains the [Queue Manager].[GUID] of the [Queue Manager] that first accepted the [Message] from the client. This value is assigned by **MSMQMessage4::Send**.
- [Body]: A **VARIANT** that contains the client-provided message payload. Clients of this protocol MAY use this field for any application-specific purpose. The server MUST not interpret or modify this field, except to encrypt the value during the message transfer process as specified by [Privacy Level].
- [Administration Queue Format Name]: A client-provided format name string that identifies an [Application Queue] to which the message transfer process will send administrative acknowledgment messages. Administrative acknowledgment messages are generated by the message transfer process to indicate the delivery outcome of a [Message] sent by a client of this protocol. Refer to the **MQMSGCLASS** enumeration for the types of administrative messages and the specific conditions in which they are produced by the message queuing system.

- [Correlation Identifier]: A client-provided OBJECTID, as specified in [\[MS-MQMQ\]](#) section 2.2.8. Clients of this protocol MAY use this field for any application-specific purpose. The server MUST not interpret or modify this field.
- [Acknowledgements Requested]: A numeric value containing bitflags that indicate the classifications of administrative acknowledgment messages to be generated by the message transfer process while delivering the [Message]. The defined values are contained in the [MQMSGACKNOWLEDGEMENT \(section 2.2.2.11\)](#) enumeration.
- [Label]: A client-provided text string. Clients of this protocol MAY use this field for any application-specific purpose. The server MUST NOT interpret or modify this field. The [Label] MUST NOT exceed 249 characters in length.
- [Time To Reach Queue]: A time span value that indicates when the message will expire relative to the [Sent Time] while it is en route to the destination [Application Queue].
- [Time To Be Received]: A time span value that indicates when the message will expire relative to the [Sent Time] after it arrives in the destination [Application Queue], and while it is waiting to be received by a client of this protocol.
- [Hash Algorithm]: An enumerated value that indicates the algorithm used to generate the hash for signing the [Message]. The defined values are contained in the [MQCALG \(section 2.2.2.18\)](#) enumeration.
-
- [Encryption Algorithm]: A numeric value that indicates the algorithm employed by the message transfer process to encrypt or decrypt the [Body]. This value is not interpreted by this protocol; rather, the message transfer process interprets and validates the value. Potential values are contained in the **MQCALG** (section 2.2.2.18) enumeration which assigns numeric values to specific cryptographic algorithms. The message transfer implementation MAY support a subset of the encryption algorithms defined by **MQCALG** (section 2.2.2.18). [<13>](#)
- [Sent Time]: A UTC date/time value that indicates when the [Message] was sent. This value is assigned by **MSMQMessage4::Send**.
- [Destination Queue Format Name]: A format name string that indicates the destination for the [Message]. This value is assigned by **MSMQMessage4::Send**.
- [Sender Certificate]: If the [Authentication Level] indicates that the [Message] was signed, this field contains an X.509 certificate [\[RFC3280\]](#) for the user identified by [Sender Identifier]. The value MAY be assigned by a client of this protocol, or by **MSMQMessage4::Send**.
- [Sender Identifier Type]: An enumerated value that indicates the format of the [Sender Identifier]. The defined values are contained in the [MQMSGSENDERIDTYPE](#) enumeration.
- [Sender Identifier]: An array of bytes that contains the security identifier (SID) of the user that sent the [Message]. If [Authentication Level] indicates that the [Message] was signed, clients of this protocol MAY consider this value to be authentic. [Sender Identifier] is assigned by **MSMQMessage4::Send** if [Sender Identifier Type] equals MQMSG_SENDERID_TYPE_SID.
- [Extension]: An array of bytes that contains a secondary client-provided message payload. Clients of this protocol MAY use this field for any application-specific purpose. The server MUST not interpret or modify this field. [Extension] is never encrypted during transfer, regardless of [Privacy Level].

- [Connector Type GUID]: A GUID that, when specified, indicates that the [Message] was sent by a **connector application**. The specific GUID value is ignored by the server, and no standard values are defined.
- [Transaction Status Queue Format Name]: A format name string that identifies the location where an OrderAck packet indicating the in-order arrival of the [Message] is to be sent by the message transfer process, as described in [\[MS-MQOB\]](#) section 3.1.5.8.6. This value is read-only for clients of this protocol, and the value is assigned by the message queuing system during the message transfer process. Beyond providing the value to clients of this protocol, the server MUST ignore this field.
- [Symmetric Key]: A **symmetric key** of the type specified by [Encryption Algorithm], encrypted such that it can only be read by the [Queue Manager] that owns the destination [Application Queue]. The [Symmetric Key], prior to being encrypted, is used to encrypt the [Body]. This value is typically assigned by the message transfer process, although it MAY be assigned by a connector application.
- [Signature]: If the [Authentication Level] indicates that the [Message] was signed, this field contains the hash specified by [Hash Algorithm], signed with the **private key** corresponding to the [Sender Certificate]. This value is typically assigned by **MSMQMessage4::Send**, although it MAY be assigned by a connector application.
- [Authentication Provider Type]: A numeric value that indicates the type of cryptographic provider used to sign the [Message]. This value is typically assigned by **MSMQMessage4::Send**, in which case only one standard value is defined: 0x1. Alternatively, this field MAY be assigned by a connector application, in which case the field is application-specific, and the server MUST ignore the value.
- [Authentication Provider Name]: A string value that indicates the name of the cryptographic provider used to sign the [Message]. This value is typically assigned by **MSMQMessage4::Send**, in which case two standard values are defined as follows:
 - "Microsoft Base Cryptographic Provider, v1.0"
 - "Microsoft Enhanced Cryptographic Provider, v1.0"

Alternatively, this field MAY be assigned by a connector application, in which case the field is application-specific, and the server MUST ignore the value.

- [Transactional Message Sequence Identifier]: A transactional message sequence identifier, as defined in [\[MS-MQOB\]](#) section 3.1.1.3. For non-transactional [Message]s, this field is empty. This value is typically assigned by the message queuing system during the message transfer process, although it MAY be assigned by a connector application.
- [IsFirstInTransaction]: A Boolean value that, when True, indicates that the [Message] is the first in a set of [Message]s with identical [Transactional Message Sequence Identifier]s. Otherwise, the value is False.
- [IsLastInTransaction]: A Boolean value that, when True, indicates that the [Message] is the last in a set of [Message]s with identical [Transactional Message Sequence Identifier]s. Otherwise, the value is False.
- [SOAP Envelope]: A string value that contains the **SOAP envelope** for an SRMP message. This value is assigned by the message transfer process and is read-only for clients of this protocol. Beyond providing the value to clients of this protocol, the server MUST ignore this field.

- [SOAP Compound Message]: A string value that contains the entire contents of a SRMP message, including both the SOAP envelope and **SOAP attachments** associated with it. This value is assigned by the message transfer process and is read-only for clients of this protocol. Beyond providing the value to clients of this protocol, the server **MUST** ignore this field.

3.1.1.24 Open Queues Table

An [Open Queues Table] **MUST** contain an associative list of zero or more [Open Queue]s indexed by [Open Queue].[Handle].

3.1.1.25 Open Queue

An [Open Queue] represents a [Queue Manager]'s granted permission to perform particular operations on a [Queue].

An [Open Queue] **MUST** contain the following elements:

- [Handle]: A numeric value that uniquely identifies the [Open Queue] within the scope of the [Open Queues Table].
- [Queue Reference]: A reference to the [Queue] for which the [Open Queue] grants access.
- [Access Mode]: An enumerated value that indicates the permitted operations granted for the [Open Queue]. The values for this field are defined by the [MQACCESS \(section 2.2.2.3\)](#) enumeration.
- [IsExclusive]: A Boolean value that, when True, indicates that the access granted to the [Queue] is exclusive. Exclusive access is granted only when no other [Open Queue]s reference the same [Queue]. Additionally, exclusive access prevents new [Open Queue]s from being created if they reference the same [Queue].

3.1.1.26 Enlisted Transactions Table

An [Enlisted Transactions Table] **MUST** contain an associative list of zero or more [Enlisted Transaction]s indexed by [Enlisted Transaction].[Transaction Identifier].

3.1.1.27 Enlisted Transaction

An [Enlisted Transaction] represents a transactional unit of work in which the [Queue Manager] will participate. Each [Enlisted Transaction] contains a set of zero or more [Transactional Operation]s that represent the individual operations associated with the [Enlisted Transaction].

An [Enlisted Transaction] **MUST** contain the following elements:

- [Transaction Identifier]: An array of sixteen (16) bytes (functionally a GUID) that uniquely identifies the [Enlisted Transaction].
- [Transactional Operations List]: A first-in, first-out (FIFO)-ordered list of zero or more [Transactional Operation]s. The contents of the list are processed by the [Transaction Commit] and [Transaction Abort] events, as defined in sections [3.1.6.1](#) and [3.1.6.2](#), respectively.

3.1.1.28 Transactional Operation

A [Transactional Operation] represents an individual message operation in the scope of an [Enlisted Transaction].

A [Transactional Operation] MUST contain the following elements:

- [Operation Type]: An enumerated value that indicates the type of operation represented by the [Transactional Operation]. The following values are defined:

Symbolic name	Meaning
Send	The [Transactional Operation] was created by IMSMQMessage::Send .
Receive	<p>The [Transactional Operation] was created by one of the following methods of MSMQQueue:</p> <ul style="list-style-type: none">▪ IMSMQQueue4::Receive_v1▪ IMSMQQueue4::ReceiveCurrent_v1▪ IMSMQQueue4::Receive▪ IMSMQQueue4::ReceiveCurrent▪ IMSMQQueue4::ReceiveByLookupId▪ IMSMQQueue4::ReceiveNextByLookupId▪ IMSMQQueue4::ReceivePreviousByLookupId▪ IMSMQQueue4::ReceiveFirstByLookupId▪ IMSMQQueue4::ReceiveLastByLookupId▪ IMSMQQueue4::ReceiveByLookupIdAllowPeek

- [Message Reference]: A reference to the [Message] that is locked by the [Transactional Operation].

3.1.2 Timers

3.1.2.1 Time To Reach Queue

An expiration time defined by the [Message].[Time To Reach Queue] data element, described in section [3.1.1.23](#). The timer is enabled if the property value is not zero or infinity. Upon expiration, the server MUST perform the behaviors defined for the [Time To Reach Queue Expiration \(section 3.1.5.1\)](#) timer event.

3.1.2.2 Time To Be Received

An expiration time defined by the [Message].[Time To Be Received] data element, described in section [3.1.1.23](#). The timer is enabled if the property value is not zero or infinity. Upon expiration, the server MUST perform the behaviors defined for the [Time To Be Received Expiration \(section 3.1.5.2\)](#) timer event.

3.1.3 Initialization

An implementation of the message queuing system **MUST** be initialized and available, featuring the system data model described here, in order for the server to respond to clients according to this protocol.

3.1.4 Message Processing Events and Sequencing Rules

The following message processing events and sequencing rules apply to the methods described in sections [3.2](#) to [3.17](#) in general.

3.1.4.1 Security

For all methods listed below, before processing the method, the server **SHOULD** obtain the identity and authorization information about the client from the underlying DCOM or RPC runtime in order to verify that the client has sufficient permissions to perform the requested operation.

3.1.4.2 Optional Arguments

Methods of the objects that are defined here utilize the Optional Arguments feature specified in [\[MS-OAUT\]](#) section 3.1.4.4.3.

For optional arguments where the client does not specify a value, the client **MUST** set the "vt" VARIANT field to VT_ERROR, and place DISP_E_PARAMNOTFOUND (0x80020004) in the "scode" VARIANT field.

3.1.4.3 Out-Parameters and Errors

Unless otherwise specified below, all methods **MUST** return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

Furthermore, out-parameter values **MUST** be ignored by the client if the server returns an error HRESULT.

3.1.5 Timer Events

The following timer events are defined for the system abstract data model. A timer **MUST** be maintained for each instance of the following time values, where the expiration time is not defined to be zero or infinite. The events are processed by the server in response to timer expirations defined in section [3.1.2](#).

3.1.5.1 Time To Reach Queue Expiration

The Time To Reach Queue Expiration event occurs when the [Time To Reach Queue](#) timer for a [Message] expires while the [Message] is in an [Outgoing Queue] pending delivery by the message transfer process. The event notification **MUST** specify the expired [Message] for which the Time To Reach Queue timer expired.

When responding to the event notification defined above, the message queuing system **MUST** follow these guidelines:

- If the [IsLocked] property of the expired [Message] equals True, take no action in response to this event notification, and postpone evaluating this event until the [Transaction Commit] event causes [IsLocked] to be set to False.

- If the [Journaling Requested] property of the expired [Message] has the MQMSG_DEADLETTER (0x1) bit set, add the [Message] to the [System Deadletter Queue] of the [Local Queue Manager].
- Finally, delete the expired [Message] from the [Outgoing Queue].

3.1.5.2 Time To Be Received Expiration

The Time To Be Received Expiration event occurs when the [Time To Be Received](#) timer for a [Message] expires while the [Message] is in an [Application Queue]. The event notification MUST specify the expired [Message] for which the [Time To Reach Queue](#) timer expired.

When responding to the event notification defined above, the message queuing system MUST follow these guidelines:

- If the [IsLocked] property of the expired [Message] equals True, take no action in response to this event, and postpone evaluating this event until the [Transaction Abort] event causes the [IsLocked] property to be set to False.
- If the [Journaling Requested] property of the expired [Message] has the MQMSG_DEADLETTER (0x1) bit set:
 - If the [IsTransactional] property of the [Queue] that contains the expired [Message] is True, add the expired [Message] to the [System Deadletter Queue] of the [Queue Manager] that accepted the [Message] from the client.
 - Else, if [IsTransactional] is False, add the expired [Message] to the [System Deadletter Queue] of the [Local Queue Manager].
- Finally, delete the expired [Message] from the [Application Queue].

3.1.6 Other Local Events

The following local events trigger operations on the [Local Queue Manager]:

- [Transaction Commit]
- [Transaction Abort]

3.1.6.1 Transaction Commit

The [Transaction Commit] event is received when a transaction manager notifies the server of the outcome of a transaction in which the server has been enlisted. The event notification MUST specify the transaction identifier for the transaction that is being committed. For more details, see [\[MS-DTCOL\]](#) section 3.5.4.5.

When processing this event, the server MUST follow these guidelines:

- Look up the [Enlisted Transaction] in the [Enlisted Transactions Table] of the [Local Queue Manager], where the [Transaction Identifier] of the [Enlisted Transaction] matches the transaction identifier input parameter of the event.
 - If the [Enlisted Transaction] is not found, take no further action.
- For each [Transactional Operation] entry in the [Transactional Operations List] of the identified [Enlisted Transaction]:

- Identify the [Message] (referred to here as *iMessage*) as the [Message] that is identified by the [Message Reference] property of the [Transactional Operation].
- If the [Operation Type] property of the [Transactional Operation] is "Send":
 - Set the [IsLocked] property of *iMessage* to False.
- Else, if the [Operation Type] property of the [Transactional Operation] is "Receive":
 - If the [Queue] that owns *iMessage* is an [Application Queue], and the [IsTargetJournalingEnabled] property is True:
 - Set the [IsLocked] property of *iMessage* to False.
 - Add *iMessage* to the [Messages Table] of the [Application Queue].[Journal Queue].
 - Delete *iMessage* from the [Messages Table] of the owning [Queue].
- Remove the identified [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager].

3.1.6.2 Transaction Abort

The [Transaction Abort] event is received when a transaction manager notifies the server of the outcome of a transaction in which the server has been enlisted. The event notification **MUST** specify the transaction identifier for the transaction that is being committed. For more details, see [\[MS-DTCO\]](#) section 3.5.4.5.

When processing this event, the server **MUST** follow these guidelines:

- Look up the [Enlisted Transaction] in the [Enlisted Transactions Table] of the [Local Queue Manager], where the [Transaction Identifier] of the [Enlisted Transaction] matches the transaction identifier input parameter of the event.
 - If the [Enlisted Transaction] is not found, take no further action.
- For each [Transactional Operation] entry in the [Transactional Operations List] of the identified [Enlisted Transaction]:
 - If the [Operation Type] property of the [Transactional Operation] is "Receive":
 - Set the [IsLocked] property of the [Message] identified by the [Message Reference] property of the [Transactional Operation] to False.
 - Else, if the [Operation Type] property of the [Transactional Operation] is "Send":
 - Delete the [Message] that is identified by the [Message Reference] property of the [Transactional Operation] from the [Messages Table] of the owning [Queue].
- Remove the identified [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager].

3.2 MSMQApplication Cclass

The MSMQApplication cclass represents either the [Queue Manager] running on the specified computer, or the [Local Queue Manager] if no **computer name** is specified. The [Queue Manager] represented by this object is referred to from here on as the represented [Queue Manager].

The MSMQApplication object provides methods and properties that can be used to obtain information about and manage the represented [Queue Manager]. This object can be used to do the following:

- Obtain the [Computer Name] or [GUID] of the represented [Queue Manager].
- Obtain management information about the represented [Queue Manager].
- Connect or disconnect the represented [Queue Manager] from the network and the [Directory].
- Register a [Directory User] with a user certificate in the [Directory]. This user certificate can be used to verify the sender for messages requesting authentication, and to ensure message integrity.
- Clean up empty message files of the represented [Queue Manager].

3.2.1 Abstract Data Model

An implementation of the [MSMQApplication](#) coclass maintains the following abstract data element:

- *ComputerName*: This instance variable specifies the string representation of the computer name on which the [Queue Manager] is running, if one is explicitly specified by the client. If not set, this variable MUST contain a NULL value, which implies that the [Local Queue Manager] will be represented by this object.

3.2.2 Timers

No protocol timers are required.

3.2.3 Initialization

- The *ComputerName* instance variable MUST be initialized to NULL value.
- Clients MAY set the *ComputerName* instance variable by invoking the [put MSMQApplication::MachineName](#) method on the object.

3.2.4 Message Processing Events and Sequencing Rules

This coclass includes three interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQApplication 3	IMSMQApplication 2	IMSMQApplication 1
7	MachineIdOfMachineName	X	X	X
8	RegisterCertificate	X	X	
9	MachineNameOfMachineId	X	X	
10	get MSMQVersionMajor	X	X	

Opnum	Method name (in the most recent interface revision)	IMSMQApplication 3	IMSMQApplication 2	IMSMQApplication n
11	get MSMQVersionMinor	X	X	
12	get MSMQVersionBuild	X	X	
13	get IsDsEnabled	X	X	
14	get Properties	X	X	
15	ActiveQueues	X		
16	get PrivateQueues	X		
17	get DirectoryServiceServer	X		
18	get IsConnected	X		
19	get BytesInAllQueues	X		
20	put Machine	X		
21	get Machine	X		
22	Connect	X		
23	Disconnect	X		
24	Tidy	X		

3.2.4.1 IMSMQApplication Interface

The **IMSMQApplication** provides methods that return information about the [Directory Queue Manager] for a given machine. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e086-dccd-11d0-aa4b-0060970debae} (coclass MSMQApplication as specified in section 1.9), which implements the IMSMQApplication interface using the UUID {D7D6E085-DCCD-11d0-AA4B-0060970DEBAE}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface, as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**. Opnums 3 through 6 are inherited from the IDispatch interface, as specified in [\[MS-OAUT\]](#) section **3.1.4**.

Methods in RPC Opnum Order

Method	Description
MachineIdOfMachineName (section 3.2.4.1.1)	Returns a string containing the GUID of the queue manager that is identified by the computer name passed in as the input parameter. Opnum: 7

3.2.4.1.1 MachineIdOfMachineName (Opnum 7)

The **MachineIdOfMachineName** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string containing the [GUID] property of the [Directory Queue Manager] identified by the computer name that was passed in as the input parameter.

```
HRESULT MachineIdOfMachineName(
    [in] BSTR MachineName,
    [out, retval] BSTR* pbstrGuid
);
```

MachineName: A [BSTR](#) that specifies the **NETBIOS** or DNS computer name for which a [GUID](#) is to be retrieved.

If this input parameter is NULL, then the [Computer Name] of the [Local Queue Manager] MUST be used as the value of this input parameter.

pbstrGuid: A pointer to a **BSTR**, that upon successful completion contains the string representation of the specified machine **GUID**. The server MUST NOT include the surrounding curly braces ({}), with the returned **GUID**. The string MUST adhere to the following format, specified using **Augmented Backus-Naur Form (ABNF)**.

```
guid          = dword-part "-" word-part "-" word-part "-"
                2byte-part "-" 6byte-part
dword-part    = 2word-part
word-part     = 2byte-part
byte-part     = 2hex-digit
hex-digit     = %x30-39 / %x41-46 / %x61-66
```

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<14>](#)

When processing this call, the server MUST follow these guidelines:

- Look up the [Directory Queue Manager] in the [Directory Queue Managers Table] of the [Directory] where the [Computer Name] property of [Directory Queue Manager] matches the MachineName input parameter.
- The server MUST return an error HRESULT if the [Directory Queue Manager] is not found, and take no further action.

- The pbstrGuid output variable MUST be set to the [GUID] property of the identified [Directory Queue Manager]; it MUST NOT contain the {} around the **GUID** value.

3.2.4.2 IMSMQApplication2 Interface

The **IMSMQApplication2** interface provides methods that return information about the queue manager on a given server. **IMSMQApplication2** inherits opnums 0 through 7 from the [IMSMQApplication \(section 3.2.4.1\)](#) interface. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e086-dccd-11d0-aa4b-0060970debae} (coclass MSMQApplication as specified in section 1.9), which implements the IMSMQApplication2 interface using the UUID {12a30900-7300-11d2-b0e6-00e02c074f6b}.

Methods in RPC Opnum Order

Method	Description
RegisterCertificate (section 3.2.4.2.1)	Registers an internal or external certificate in the directory service for the user invoking the method. Opnum: 8
MachineNameOfMachineId (section 3.2.4.2.2)	Returns the computer name of the queue manager identified by the GUID that was passed in as the input parameter. Opnum: 9
MSMQVersionMajor (section 3.2.4.2.3), get MSMQVersionMajor	Returns the major version number of the server. Opnum: 10
MSMQVersionMinor (section 3.2.4.2.4), get MSMQVersionMinor	Returns the minor version number of the server. Opnum: 11
MSMQVersionBuild (section 3.2.4.2.5), get MSMQVersionBuild	Returns the build version number of the server. Opnum: 12
IsDsEnabled (section 3.2.4.2.6), get IsEnabled	Returns a Boolean value that indicates whether the queue manager represented by this object is configured to use the directory service. Opnum: 13
Properties (section 3.2.4.2.7), get Properties	This method is not implemented. Opnum: 14

3.2.4.2.1 RegisterCertificate (Opnum 8)

The **RegisterCertificate** method registers an [Internal Certificate] or an [External Certificate] in the [Directory] for the [Directory User]. Implementations of this protocol can use the certificate to verify the sender for messages that are requesting authentication, and to ensure message integrity.

```
HRESULT RegisterCertificate(
    [in, optional] VARIANT* Flags,
    [in, optional] VARIANT* ExternalCertificate
);
```

Flags: A pointer to a VARIANT that contains a VT_I4 integer, that corresponds to the MQCERT_REGISTER enumeration as defined below.

Value	Meaning
MQCERT_REGISTER_ALWAYS 0x00000001	Register either an [Internal Certificate] or [External Certificate] in the [Directory] for the [Directory User]. If the <i>ExternalCertificate</i> input parameter is not specified or is NULL, then the server MUST delete any existing [Internal Certificate] from the internal store, and unregister the existing [Internal Certificate] from the [Directory User]. The server MUST register a newly created [Internal Certificate] in the [Directory] for the [Directory User] and add it to the internal store. If the <i>ExternalCertificate</i> is not NULL, then the server MUST register the [External Certificate], as specified by the <i>ExternalCertificate</i> input parameter, in the [Directory] for the [Directory User].
MQCERT_REGISTER_IF_NOT_EXIST 0x00000002	Register an [Internal Certificate] in the [Directory] for the [Directory User] only if it does not exist in the internal certificate store. This option cannot be used when registering an [External Certificate].

If not specified by the client, the server MUST use the default value MQCERT_REGISTER_ALWAYS (0x00000001) in place of the unspecified value.

ExternalCertificate: A pointer to a VARIANT that contains a byte array (VT_ARRAY|VT_UI1), or a pointer (VT_BYREF) to a byte array, that specifies the binary representation of the [External Certificate] that is to be registered. The [External Certificate] MUST contain an X.509-encoded certificate, as specified in [\[RFC3280\]](#).

If not specified by the client, the server MUST register an [Internal Certificate].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<15>](#)

When processing this call, the server MUST follow these guidelines:

- If the *ComputerName* instance variable is not NULL:
 - The server MUST return E_NOTIMPL (0x80004001) and take no further action.
- If the *Flags* input parameter is equal to MQCERT_REGISTER_IF_NOT_EXIST, and the *ExternalCertificate* input parameter is not NULL :
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006) and take no further action.
- The server MUST identify the [Directory User], referred to here as *iDirectoryUser*, as follows:
 - Look up the [Directory User] in the [Directory Users Table] of the [Directory], where the [Security Descriptor] of the [Directory User] matches the SID [<16>](#) of the user invoking the method.
 - The server MUST return an error HRESULT if the [Directory User] is not found, and take no further action.

- If the *ExternalCertificate* input parameter is specified and is not NULL:
 - The server MUST add a new [External Certificate] to the [Certificates Table] of the *iDirectoryUser*, and set:
 - The [X509] property to the X.509-encoded certificate BLOB of the *ExternalCertificate* input parameter.
 - The [Digest] property to the 16-byte output of the MD5 algorithm, as specified in [\[RFC1321\]](#); applied to the *ExternalCertificate* input parameter.
- Else:
 - The server MUST identify the [Internal Certificate], referred to here as *iInternalCertificate*, in the internal certificate store on the server computer.
 - If *iInternalCertificate* is found:
 - If the *Flags* input parameter is equal to MQCERT_REGISTER_IF_NOT_EXIST:
 - The server MUST return MQ_INFORMATION_INTERNAL_USER_CERT_EXIST (0x400E000A) and take no further action.
 - Else:
 - The server MUST remove the [Internal Certificate] from the [Certificates Table] of the *iDirectoryUser*, where the [Digest] of the [Internal Certificate] matches the [Digest] of the *iInternalCertificate*.
 - The server MUST delete the *iInternalCertificate* from the internal certificate store.
 - The server MUST create an X.509-encoded certificate BLOB, as specified in [\[RFC3280\]](#), and add the certificate to the internal certificate store on the server computer.
 - The server MUST add a new [Internal Certificate] to the [Certificates Table] of the *iDirectoryUser* and set:
 - The [X509] property to the created X.509 certificate BLOB.
 - The [Digest] property to the 16-byte output of the MD5 algorithm, as specified in [\[RFC1321\]](#).

3.2.4.2.2 MachineNameOfMachineId (Opnum 9)

The **MachineNameOfMachineId** method is received by the server in an RPC_REQUEST packet. In response, the server returns a string containing the [Computer Name] of the [Directory Queue Manager] identified by the [GUID](#) that was passed in as the input parameter.

```
HRESULT MachineNameOfMachineId(
    [in] BSTR bstrGuid,
    [out, retval] BSTR* pbstrMachineName
);
```

bstrGuid: A [BSTR](#) that specifies an identifier for the computer name, in the **GUID** format. The server MUST accept **GUIDs** with or without surrounding curly braces {}.

pbstrMachineName: A pointer to a **BSTR**, that upon successful completion contains the computer name in the DNS or Universal Naming Convention (UNC) name format.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<17>](#)

When processing this call, the server MUST follow these guidelines:

- Look up the [Directory Queue Manager] in the [Directory Queue Managers Table] of the [Directory], where the [GUID] of the [Directory Queue Manager] matches the *bstrGuid* input parameter.
- The server MUST return an error HRESULT if the [Directory Queue Manager] is not found, and take no further action.
- The *pbstrMachineName* output variable MUST be set to the [Computer Name] of the identified [Directory Queue Manager].

3.2.4.2.3 MSMQVersionMajor (Opnum 10)

The **MSMQVersionMajor** method is received by the server in an RPC_REQUEST packet. In response, the server returns the major version number of the server.

```
[propget] HRESULT MSMQVersionMajor(  
    [out, retval] short* psMSMQVersionMajor  
);
```

psMSMQVersionMajor: A pointer to a short that upon successful completion contains the major version number of the server.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ComputerName* instance variable is not NULL:
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006), and take no further action.
- The *psMSMQVersionMajor* output variable MUST be set to the major version number [<18>](#) of the server.

3.2.4.2.4 MSMQVersionMinor (Opnum 11)

The **MSMQVersionMinor** method is received by the server in an RPC_REQUEST packet. In response, the server returns the minor version number of the server.

```
[propget] HRESULT MSMQVersionMinor(  
    [out, retval] short* psMSMQVersionMinor  
);
```

psMSMQVersionMinor: A pointer to a short, that upon successful completion contains the minor version number of the server.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ComputerName* instance variable is not NULL:
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006), and take no further action.
- The *psMSMQVersionMinor* output variable MUST be set to the minor version number [<19>](#) of the server.

3.2.4.2.5 MSMQVersionBuild (Opnum 12)

The **MSMQVersionBuild** method is received by the server in an RPC_REQUEST packet. In response, the server returns the build version number of the server.

```
[propget] HRESULT MSMQVersionBuild(  
    [out, retval] short* psMSMQVersionBuild  
);
```

psMSMQVersionBuild: A pointer to a short, that upon successful completion contains the build version number of the server.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ComputerName* instance variable is not NULL:
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006), and take no further action.
- The *psMSMQVersionBuild* output variable MUST be set to the build version number [<20>](#) of the server.

3.2.4.2.6 IsDsEnabled (Opnum 13)

The **IsDsEnabled** method is received by the server in an RPC_REQUEST packet. In response, the server indicates whether the [Local Queue Manager] is configured to use the [Directory].

```
[propget] HRESULT IsDsEnabled(  
    [out, retval] VARIANT_BOOL* pfIsDsEnabled  
);
```

pfIsDsEnabled: A pointer to a VARIANT_BOOL, that upon successful completion contains one of the following values.

Value	Meaning
VARIANT_TRUE 0xffff	The [Local Queue Manager] is configured to use the [Directory].

Value	Meaning
VARIANT_FALSE 0x0000	The [Local Queue Manager] is not configured to use the [Directory].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ComputerName* instance variable is not NULL:
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006), and take no further action.
- The *pfIsDsEnabled* output variable MUST be set to the [IsDirectoryEnabled] value of the [Local Queue Manager].

3.2.4.2.7 Properties (Opnum 14)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(
    [out, retval] IDispatch** ppcolProperties
);
```

ppcolProperties: A pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The server MUST take no action and return E_NOTIMPL (0x80004001).

3.2.4.3 IMSMQApplication3 Interface

The **IMSMQApplication3** interface provides methods that allow clients to administer queues. **IMSMQApplication3** inherits opnums 0 through 14 from the [IMSMQApplication2 \(section 3.2.4.2\)](#) interface. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e086-dccd-11d0-aa4b-0060970debae} (coclass MSMQApplication as specified in section 1.9), which implements the IMSMQApplication3 interface using the UUID {eba96b1f-2168-11d3-898c-00e02c074f6b}.

Methods in RPC Opnum Order

Method	Description
ActiveQueues (section 3.2.4.3.1)	Returns the format names of all the active queues of the queue manager represented by this object. Opnum: 15
get PrivateQueues (section 3.2.4.3.2)	Returns the path names of all the private queues of the queue manager represented by this object. Opnum: 16

Method	Description
get DirectoryServiceServer (section 3.2.4.3.3)	Returns the name of the directory service server. Opnum: 17
get IsConnected (section 3.2.4.3.4)	Returns a Boolean value that indicates the connection status of the queue manager represented by this object. Opnum: 18
get BytesInAllQueues (section 3.2.4.3.5)	Returns the number of message bytes currently stored in all queues of the queue manager represented by this object. Opnum: 19
put Machine (section 3.2.4.3.6)	Sets the name of the computer on which the queue manager is to be found. Opnum: 20
get Machine (section 3.2.4.3.7)	Returns the name of the computer on which the queue manager represented by this object is running. Opnum: 21
Connect (section 3.2.4.3.8)	Connects the queue manager represented by this object to the network and to the directory service. Opnum: 22
Disconnect (section 3.2.4.3.9)	Disconnects the queue manager represented by this object from the network and the directory service. Opnum: 23
Tidy (section 3.2.4.3.10)	Cleans up empty message files of the queue manager that is represented by this object. Opnum: 24

3.2.4.3.1 ActiveQueues (Opnum 15)

The **ActiveQueues** method returns an array of strings that contains the format names of all the [Queue]s in the [Queues Table] of the represented [Queue Manager], where the [IsActive] property of the [Queue] is True.

```
HRESULT ActiveQueues(
    [out, retval] VARIANT* pvActiveQueues
);
```

pvActiveQueues: A pointer to a VARIANT, that upon successful completion contains an array of zero or more strings (VT_ARRAY | VT_BSTR) that specify the format names of all the [Queue]s that the represented [Queue Manager] has with [IsActive] property set to True.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:

- If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager]
- Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up all the [Queue]s from the [Queues Table] of the *iQueueManager* that have the [IsActive] property set to True.
- The *pvActiveQueues* output variable MUST be set to an array that contains the format names that specify all the identified [Queue]s.

3.2.4.3.2 PrivateQueues (Opnum 16)

The **IMSMQApplication3::PrivateQueues** method returns an array of strings that contains the path names of all the [Private Queue]s in the [Queues Table] of the represented [Queue Manager].

```
[propget] HRESULT PrivateQueues(
    [out, retval] VARIANT* pvPrivateQueues
);
```

pvPrivateQueues: A pointer to a VARIANT, that upon successful completion contains an array of zero or more strings (VT_ARRAY | VT_BSTR) that specify all the private queue path names.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up all the [Private Queue]s from the [Queues Table] of the *iQueueManager*.
- The *pvPrivateQueues* output variable MUST be set to an array that contains the path names that specify all the identified [Private Queue]s.

3.2.4.3.3 DirectoryServiceServer (Opnum 17)

The **DirectoryServiceServer** method returns the name of the current [Directory] computer.

```
[propget] HRESULT DirectoryServiceServer(  
    [out, retval] BSTR* pbstrDirectoryServiceServer  
);
```

pbstrDirectoryServiceServer: A pointer to a [BSTR](#), that upon successful completion contains the name of the [Directory] computer in the DNS or NETBIOS format, prefixed by "\\". The string MUST adhere to the following format, specified using Augmented BNF.

```
DirectoryServer = "\\\" Name  
Alpha          = %x41-5A / %x61-7A  
Name           = 1*Alpha
```

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pbstrDirectoryServiceServer* output variable MUST be set to the DNS or NETBIOS name of the [Directory] computer, [21](#) prefixed by "\\".

3.2.4.3.4 IsConnected (Opnum 18)

The **IsConnected** method returns a Boolean value that indicates the connection status of the represented [Queue Manager].

```
[propget] HRESULT IsConnected(  
    [out, retval] VARIANT_BOOL* pfIsConnected  
);
```

pfIsConnected: A pointer to a VARIANT_BOOL, that upon successful completion contains one of the following values.

Value	Meaning
VARIANT_TRUE (0xffff)	The represented [Queue Manager] is connected to the network and the [Directory].
VARIANT_FALSE (0x0000)	The represented [Queue Manager] is disconnected from the network and the [Directory].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:

- Find the [Local Queue Manager].
- Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
- The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- The *pfIsConnected* output variable MUST be set to the [IsConnected] property of the *iQueueManager*.

3.2.4.3.5 BytesInAllQueues (Opnum 19)

The **BytesInAllQueues** method returns the number of message bytes currently stored in all [Queue]s in the [Queues Table] of the represented [Queue Manager].

```
[propget] HRESULT BytesInAllQueues (
    [out, retval] VARIANT* pvBytesInAllQueues
);
```

pvBytesInAllQueues: A pointer to a VARIANT, that upon successful completion contains a 64-bit integer (VT_I8) that specifies (in bytes) the amount of data stored in all [Queue]s in the [Queues Table] of the [Queue Manager] on the specified computer, or of the [Local Queue Manager].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- The *pvBytesInAllQueues* output variable MUST be set to the sum of all the [Total Bytes] properties of all the [Queue]s in the [Queues Table] of the *iQueueManager*.

3.2.4.3.6 Machine (Opnum 20)

The **Machine** method sets the *ComputerName* instance variable of the object.

```
[propput] HRESULT Machine (
    [in] BSTR bstrMachine
```

```
);
```

bstrMachine: A BSTR that specifies the computer name in the DNS or NETBIOS format.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

The server MUST set the *ComputerName* instance variable to the value of the *bstrMachine* input parameter.

3.2.4.3.7 Machine (Opnum 21)

The **Machine** method returns the [Computer Name] of the represented [Queue Manager].

```
[propget] HRESULT Machine(  
    [out, retval] BSTR* pbstrMachine  
);
```

pbstrMachine: Pointer to a BSTR that upon successful completion contains the computer name that this object refers to in the DNS or NETBIOS format.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ComputerName* instance variable is NULL:
 - The pbstrMachine output variable MUST be set to the [Computer Name] of the [Local Queue Manager].
- Else:
 - The pbstrMachine output variable MUST be set to the *ComputerName* instance variable.

3.2.4.3.8 Connect (Opnum 22)

The **Connect** method connects the represented [Queue Manager] to the network and to the [Directory].

```
HRESULT Connect();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<22>](#)

When processing this call, the server MUST follow these guidelines:

- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:

- Find the [Local Queue Manager].
- Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- The *iQueueManager* MUST attempt to connect to the network and the [Directory].
- If the connection attempt is successful:
 - The server MUST set the [IsConnected] property of the *iQueueManager* to True.
- Else:
 - The server MUST return an error HRESULT and take no further action.

3.2.4.3.9 Disconnect (Opnum 23)

The **Disconnect** method disconnects the represented [Queue Manager] from the network and the [Directory].

```
HRESULT Disconnect();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<23>](#)

When processing this call, the server MUST follow these guidelines:

- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- The *iQueueManager* MUST attempt to disconnect from the network and the [Directory].
- If the attempt to disconnect is successful:
 - The server MUST set the [IsConnected] property of the *iQueueManager* to False.
- Else:
 - The server MUST return an error HRESULT and take no further action.

3.2.4.3.10 Tidy (Opnum 24)

The **Tidy** method is received by the server in an RPC_REQUEST packet. In response, the server cleans up the empty message files of the represented [Queue Manager].

```
HRESULT Tidy();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<24>](#24)

When processing this call, the server MUST follow these guidelines:

- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- The *iQueueManager* MUST attempt to clean up the empty message files.

3.2.5 Timer Events

No timer events are required.

3.2.6 Other Local Events

No local events are required.

3.3 MSMQManagement Coclass

An implementation of the MSMQManagement coclass represents the state of a [Queue] with the [IsActive] property set to True. This [Queue] is referred to as the represented [Queue] in this document in the description of this coclass.

This object encapsulates the administrative aspects of the represented [Queue].

3.3.1 Abstract Data Model

An implementation of the [MSMQManagement](#) coclass maintains the following data elements:

- *ComputerName*: This instance variable specifies the string representation of the computer name on which the [Queue Manager] that owns the represented [Queue] is running, if one is explicitly specified by the client. If not set, this variable MUST contain a NULL value, which indicates that the [Local Queue Manager] owns the [Queue].

- *QueueFormatName*: This instance variable specifies the string representation of the format name that describes the represented [Queue].
- *ObjectIsInitialized*: A Boolean flag that is set to True if the object has been successfully initialized by invoking the Init method.

An implementation of the MSMQManagement coclass SHOULD also provide the implementation for the [IMSMQQueueManagement](#) and [IMSMQOutgoingQueueManagement](#) interfaces.

On invoking [IUnknown::QueryInterface](#) (refer to section 3.1) with the interface identifier of **IMSMQQueueManagement**, this coclass SHOULD return the **IMSMQQueueManagement** coclass if the represented [Queue] is not an [Outgoing Queue], or else return an error.

On invoking **IUnknown::QueryInterface** with the interface identifier of **IMSMQOutgoingQueueManagement**, this coclass SHOULD return the **IMSMQOutgoingQueueManagement** coclass if the represented [Queue] is an [Outgoing Queue], or else return an error.

3.3.2 Timers

No protocol timers are required.

3.3.3 Initialization

- The server MUST set the *ObjectIsInitialized* instance variable to False, and the *ComputerName* instance variable to a NULL value.
- Clients MUST initialize the [MSMQManagement](#) object by invoking the [IMSMQManagement::Init](#) method on the object.

3.3.4 Message Processing Events and Sequencing Rules

The [MSMQManagement](#) coclass defines a single interface: [IMSMQManagement](#).

3.3.4.1 IMSMQManagement Interface

The **IMSMQManagement** interface provides methods that return information about a queue. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {39ce96fe-f4c5-4484-a143-4c2d5d324229} (coclass MSMQManagement as specified in section 1.9), which implements the IMSMQManagement interface using the UUID {be5f0241-e489-4957-8cc4-a452fcf3e23e}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface, as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the IDispatch interface, as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
Init (section 3.3.4.1.1)	Initializes the object to represent the state of an active queue. Opnum: 7
FormatName (section 3.3.4.1.2)	Returns a format name for the queue represented by this

Method	Description
get FormatName	object. Opnum: 8
Machine (section 3.3.4.1.3) , get Machine	Returns the name of the computer on which the queue manager that owns the queue represented by this object is running. Opnum: 9
MessageCount (section 3.3.4.1.4) , get MessageCount	Returns the number of messages in the queue represented by this object. Opnum: 10
ForeignStatus (section 3.3.4.1.5) , get Foreign Status	Returns a flag that indicates whether the queue represented by this object is a foreign queue . Opnum: 11
QueueType (section 3.3.4.1.6) , get QueueType	Returns a flag that indicates the type of the queue represented by this object. Opnum: 12
IsLocal (section 3.3.4.1.7) , get IsLocal	Returns a Boolean value that indicates whether the queue represented by this object is owned by the local queue manager or a remote queue manager. Opnum: 13
TransactionalStatus (section 3.3.4.1.8) , get TransactionalStatus	Returns a flag that indicates whether the queue represented by this object is a transactional queue or a non-transactional queue. Opnum: 14
BytesInQueue (section 3.3.4.1.9) get, BytesInQueue	Returns the number of message bytes in the queue represented by this object. Opnum: 15

3.3.4.1.1 Init (Opnum 7)

The **Init** method is received by the server in an RPC_REQUEST packet. In response, the server initializes the object to represent the state of a [Queue]. The represented [Queue] MUST have the [IsActive] property set to True. This method MUST be called prior to calling any other operation on [MSMQManagement](#).

```
HRESULT Init(
    [in, optional] VARIANT* Machine,
    [in, optional] VARIANT* Pathname,
    [in, optional] VARIANT* FormatName
);
```

Machine: A pointer to a VARIANT that contains a [BSTR](#) that contains a string representation of a computer name in the DNS or NETBIOS format. If this parameter is not specified or is NULL, the server MUST ignore this parameter.

Pathname: A pointer to a VARIANT that contains a **BSTR** that contains a string representation of the path name describing a [Queue].

FormatName: A pointer to a VARIANT that contains a **BSTR** that contains a string representation of the format name describing a [Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *Pathname* input parameter is NULL or is not specified, AND the *FormatName* input parameter is NULL or is not specified:
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006), and take no further action.
- If the *Pathname* input parameter is not NULL, and the *FormatName* input parameter is not NULL:
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006), and take no further action.
- If the *Machine* input parameter is specified and is not NULL:
 - The server MUST set the *ComputerName* instance variable to the value of the *Machine* input parameter.
- If the *FormatName* input parameter is specified and is not NULL, and the *Pathname* input parameter is NULL:
 - The server MUST set the *QueueFormatName* instance variable to the value of the *FormatName* input parameter.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- If the *Pathname* input parameter is specified and is not NULL, and the *FormatName* input parameter is NULL:
 - Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *Pathname* input parameter.
 - The server MUST return an error HRESULT if the [Queue] cannot be identified, and take no further action.
 - The server MUST set the *QueueFormatName* instance variable to the value of the format name that specifies the identified [Queue].

- Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if the [Queue] cannot be identified, and take no further action.
- If the [IsActive] property of the identified [Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004) and take no further action.
- The server MUST set the *ObjectIsInitialized* instance variable to True.

3.3.4.1.2 FormatName (Opnum 8)

The **FormatName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the format name of the represented [Queue].

```
[propget] HRESULT FormatName(
    [out, retval] BSTR* pbstrFormatName
);
```

pbstrFormatName: A pointer to a [BSTR](#) that upon successful completion contains the format name of the represented [Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The *pbstrFormatName* output variable MUST be set to the value of the *QueueFormatName* instance variable.

3.3.4.1.3 Machine (Opnum 9)

The **Machine** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Computer Name] of the [Queue Manager] that owns the represented [Queue].

```
[propget] HRESULT Machine(
    [out, retval] BSTR* pbstrMachine
);
```

pbstrMachine: A pointer to a [BSTR](#), that upon successful completion contains the [Computer Name] of the [Queue Manager] that owns the represented [Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- If the *ComputerName* instance variable is NULL:
 - The *pbstrMachine* output variable MUST be set to the [Computer Name] of the [Local Queue Manager].
- Else:
 - The *pbstrMachine* output variable MUST be set to the value of the *ComputerName* instance variable.

3.3.4.1.4 MessageCount (Opnum 10)

The **MessageCount** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Message Count] of the [Messages Table] of the represented [Queue].

```
[propget] HRESULT MessageCount(
    [out, retval] long* plMessageCount
);
```

plMessageCount: A pointer to a long, that upon successful completion contains the number of messages in the represented [Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Queue] is False:

- The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.
- The *plMessageCount* output variable MUST be set to the [Message Count] of the [Messages Table] of the identified [Queue].

3.3.4.1.5 ForeignStatus (Opnum 11)

The **ForeignStatus** method is received by the server in an RPC_REQUEST packet. In response, the server returns an enumerated value indicating whether a [Queue] is a foreign queue, or an [Outgoing Queue] that transfers messages to a foreign queue.

```
[propget] HRESULT ForeignStatus(
    [out, retval] long* plForeignStatus
);
```

plForeignStatus: A pointer to a long that corresponds to the FOREIGN_STATUS enumeration as defined below.

Value	Meaning
MQ_STATUS_FOREIGN 0x00000000	The represented [Queue] is a foreign queue, or the represented [Queue] is an [Outgoing Queue] that transfers to a foreign queue.
MQ_STATUS_NOT_FOREIGN 0x00000001	The represented [Queue] is not a foreign queue, or the represented [Queue] is not an [Outgoing Queue] that transfers to a foreign queue.
STATUS_UNKNOWN 0x00000002	The message queuing system is unable to determine whether the represented [Queue] is a foreign queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
- The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.

- Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Queue] is found, and take no further action.
- If the identified [Queue] is a [Public Queue], and the [IsDSEnabled] property of the *iQueueManager* is False:
 - The *plForeignStatus* output variable MUST be set to STATUS_UNKNOWN.
 - The server MUST return S_OK and take no further action.
- If the [IsActive] property of the identified [Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004).
- The *plForeignStatus* output variable MUST be set to MQ_STATUS_FOREIGN if the identified [Queue] is a foreign queue, or if the identified [Queue] is an [Outgoing Queue] that transfers to a foreign queue. Otherwise, *plForeignStatus* MUST equal MQ_STATUS_NOT_FOREIGN.

3.3.4.1.6 QueueType (Opnum 12)

The **QueueType** method is received by the server in an RPC_REQUEST packet. In response, the server returns the type of the referenced [Queue].

```
[propget] HRESULT QueueType(
    [out, retval] long* plQueueType
);
```

plQueueType: A pointer to a long that corresponds to the QUEUE_TYPE enumeration as defined below.

Value	Meaning
MQ_TYPE_PUBLIC 0x00000000	The represented [Queue] is a [Public Queue].
MQ_TYPE_PRIVATE 0x00000001	The represented [Queue] is a [Private Queue].
MQ_TYPE_MACHINE 0x00000002	The represented [Queue] is a [System Queue].
MQ_TYPE_CONNECTOR 0x00000003	The represented [Queue] is a connector queue .
MQ_TYPE_MULTICAST 0x00000004	The represented [Queue] is an [Outgoing Queue] with the [Destination Format Name] specifying a multicast address.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:

- The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
- The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.
- If the identified [Queue] is an [Outgoing Queue], and the [Destination Format Name] of the identified [Outgoing Queue] specifies a multicast address:
 - The *plQueueType* output variable MUST be set to MQ_TYPE_MULTICAST.
- Else, if the identified [Queue] is a connector queue:
 - The *plQueueType* output variable MUST be set to MQ_TYPE_CONNECTOR.
- Else, if the identified [Queue] is a [System Queue]:
 - The *plQueueType* output variable MUST be set to MQ_TYPE_MACHINE.
- Else, if the identified [Queue] is a [Public Queue]:
 - The *plQueueType* output variable MUST be set to MQ_TYPE_PUBLIC.
- Else:
 - The *plQueueType* output variable MUST be set to MQ_TYPE_PRIVATE.

3.3.4.1.7 IsLocal (Opnum 13)

The **IsLocal** method is received by the server in an RPC_REQUEST packet. In response, the server returns a Boolean value that indicates whether the represented [Queue] is an [Outgoing Queue] (False) or not (True).

```
[propget] HRESULT IsLocal(
    [out, retval] VARIANT_BOOL* pfIsLocal
);
```

pfIsLocal: A pointer to a VARIANT_BOOL, that upon successful completion contains VARIANT_FALSE (0x0000) or VARIANT_TRUE (0xffff) values, depending on whether the represented [Queue] is an [Outgoing Queue] or not, respectively.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.
- If the identified [Queue] is an [Outgoing Queue]:
 - The *pfIsLocal* output variable MUST be set to False.
- Else:
 - The *pfIsLocal* output variable MUST be set to True.

3.3.4.1.8 TransactionalStatus (Opnum 14)

The **TransactionalStatus** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IsTransactional] flag of the represented [Queue].

```
[propget] HRESULT TransactionalStatus(  
    [out, retval] long* plTransactionalStatus  
);
```

plTransactionalStatus: A pointer to a long that corresponds to the XACT_STATUS enumeration as defined below.

Value	Meaning
MQ_XACT_STATUS_XACT 0x00000000	The [IsTransactional] property of the represented [Queue] is True.
MQ_XACT_STATUS_NOT_XACT 0x00000001	The [IsTransactional] property of the represented [Queue] is False.
MQ_XACT_STATUS_UNKNOWN 0x00000002	The represented [Queue] is a [Public Queue], and the [IsDSEnabled] property of the [Queue Manager] that owns the represented [Queue] is False.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Queue] is found, and take no further action.
- If the identified [Queue] is a [Public Queue], and the [IsDSEnabled] property of the *iQueueManager* is False:
 - The *pITransactionStatus* output variable MUST be set to STATUS_UNKNOWN.
 - The server MUST return S_OK and take no further action.
- If the [IsActive] property of the identified [Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004).
- The *pITransactionStatus* output variable MUST be set to the [IsTransactional] property of the identified [Queue].

3.3.4.1.9 BytesInQueue (Opnum 15)

The **BytesInQueue** method is received by the server in an `RPC_REQUEST` packet. In response, the server returns the [Total Bytes] property of the [Messages Table] of the represented [Queue].

```
[propget] HRESULT BytesInQueue(  
    [out, retval] VARIANT* pvBytesInQueue  
);
```

pvBytesInQueue: A pointer to a VARIANT, that upon successful completion contains an unsigned 64-bit integer (VT_UI8) that specifies (in bytes) the amount of data in the represented [Queue].

Return Values: The method MUST return `S_OK` (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return `MQ_ERROR_UNINITIALIZED_OBJECT` (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return with an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return with an error HRESULT if no [Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Queue] is False:
 - The server MUST return `MQ_ERROR_QUEUE_NOT_ACTIVE` (0xC00E0004), and take no further action.
- The *pvBytesInQueue* output variable MUST be set to the [Total Bytes] property of the [Messages Table] of the identified [Queue].

3.3.5 Timer Events

No timer events are required.

3.3.6 Other Local Events

No local events are required.

3.4 MSMQueueManagement Coclass

The MSMQueueManagement object represents the state of an [Application Queue], referred to here as the represented [Application Queue], with the [IsActive] property set to True. This object can be used to do the following:

- Obtain delivery information about transactional messages sent to the represented [Application Queue].
- Obtain administrative information that is specific to the represented [Application Queue].

3.4.1 Abstract Data Model

An implementation of the [MSMQManagement](#) coclass SHOULD provide the implementation for the [MSMQQueueManagement](#) coclass as well. The represented [Queue] of the MSMQueueManagement coclass is the represented [Application Queue] of this coclass. This object inherits the instance variables defined in the MSMQueueManagement coclass.

Refer to section [3.3.1](#) for the abstract data model of the MSMQueueManagement coclass.

3.4.2 Timers

No protocol timers are required.

3.4.3 Initialization

To obtain an instance of the [MSMQQueueManagement](#) object:

Clients MUST create an instance of the [MSMQManagement](#) object, and initialize it by invoking the [IMSMQManagement::Init](#) method on the object. Refer to section [3.3.3](#) for the initialization of the MSMQueueManagement coclass.

Clients MUST obtain the instance of MSMQueueManagement by invoking a call to [IUnknown::QueryInterface](#) (refer to section [3.1](#)) on the MSMQueueManagement object, with the interface identifier of [IMSMQQueueManagement](#).

If the represented [Queue] of the MSMQueueManagement object is an [Outgoing Queue], then the server MUST return E_NOINTERFACE (0x80004002).

3.4.4 Message Processing Events and Sequencing Rules

The [MSMQQueueManagement](#) coclass defines a single interface: [IMSMQQueueManagement](#).

3.4.4.1 IMSMQQueueManagement Interface

The **IMSMQQueueManagement** interface provides methods that return information about a journal queue. **IMSMQQueueManagement** inherits opnums 0 through 15 from the [IMSMQManagement \(section 3.3.4.1\)](#) interface. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {33b6d07e-f27d-42fa-b2d7-bf82e11e9374} (coclass MSMQueueManagement as

specified in section 1.9), which implements the IMSMQQueueManagement interface using the UUID {7fbe7759-5760-444d-b8a5-5e7ab9a84cce}.

Methods in RPC Opnum Order

Method	Description
JournalMessageCount (section 3.4.4.1.1) , get JournalMessageCount	Returns the number of journal messages in the journal queue that is associated with the queue represented by this object. Opnum: 16
BytesInJournal (section 3.4.4.1.2) , get BytesInJournal	Returns the number of message bytes in the journal queue that is associated with the queue represented by this object. Opnum: 17
EodGetReceiveInfo (section 3.4.4.1.3)	Returns an array of collections of Exactly Once Delivery (EOD) properties, with one collection for each queue manager that is transferring transactional messages to the queue that is represented by this object. Opnum: 18

3.4.4.1.1 JournalMessageCount (Opnum 16)

The **JournalMessageCount** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Message Count] of the [Messages Table] of the [Journal Queue] that is associated with the represented [Application Queue].

```
[propget] HRESULT JournalMessageCount(  
    [out, retval] long* plJournalMessageCount  
);
```

plJournalMessageCount: A pointer to a long, that upon successful completion contains the number of messages in the [Journal Queue] that is associated with the represented [Application Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.

- The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Application Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Application Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Application Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.
- Look up the associated [Journal Queue] using the [Journal Queue] property of the identified [Application Queue].
- If the [Journal Queue] is not found:
 - The server MUST return an error HRESULT and take no further action.
- The *plJournalMessageCount* output variable MUST be set to the [Message Count] of the [Messages Table] of the identified [Journal Queue].

3.4.4.1.2 BytesInJournal (Opnum 17)

The **BytesInJournal** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Total Bytes] property of the [Messages Table] of the [Journal Queue] that is associated with the represented [Application Queue].

```
[propget] HRESULT BytesInJournal(
    [out, retval] VARIANT* pvBytesInJournal
);
```

pvBytesInJournal: A pointer to a VARIANT, that upon successful completion contains an unsigned 64-bit integer (VT_UI8) that specifies the number of message bytes in the [Journal Queue] that is associated with the represented [Application Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:

- Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
- The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Application Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Application Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Application Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.
- Look up the associated [Journal Queue] using the [Journal Queue] property of the identified [Application Queue].
- If the [Journal Queue] is not found:
 - The server MUST return an error HRESULT and take no further action.
- The *pvBytesInJournal* output variable MUST be set to the [Total Bytes] property of the [Messages Table] of the identified [Journal Queue].

3.4.4.1.3 EodGetReceiveInfo (Opnum 18)

The **EodGetReceiveInfo** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [EODTargetInfos] property of the represented [Application Queue].

```
HRESULT EodGetReceiveInfo(
    [out, retval] VARIANT* pvCollection
);
```

pvCollection: A pointer to a VARIANT containing an array of VARIANTs (VT_ARRAY|VT_VARIANT), in which each VARIANT contains an [EODTargetInfo \(section 2.2.3.1\)](#) collection.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:

- Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
- The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Application Queue] from the [Queues Table] of the *iQueueManager*, identified by the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Application Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Application Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004,) and take no further action.
- The *pvCollection* output variable MUST be set to the [EODTargetInfos] property of the identified [Application Queue].

3.4.5 Timer Events

No timer events are required.

3.4.6 Other Local Events

No local events are required.

3.5 MSMQOutgoingQueueManagement Coclass

The MSMQOutgoingQueueManagement object represents the state of an [Outgoing Queue], referred to here as the represented [Outgoing Queue], with the [IsActive] property set to True. This object can be used to do the following:

- Obtain delivery information about transactional messages in the represented [Outgoing Queue] that are being transferred to the destination queue. For more details on the message transfer process, refer to section [1.4.1](#).
- Pause and restart the transfer of messages from the represented [Outgoing Queue].
- Obtain administrative information that is specific to the represented [Outgoing Queue].

3.5.1 Abstract Data Model

An implementation of the [MSMQManagement](#) coclass SHOULD provide the implementation for the [MSMQOutgoingQueueManagement](#) coclass as well. The represented [Queue] of the MSMQManagement coclass is the represented [Outgoing Queue] of this coclass. This object inherits the instance variables defined in the MSMQManagement coclass. Refer to section [3.3.1](#) for the abstract data model of the MSMQManagement coclass.

3.5.2 Timers

No protocol timers are required.

3.5.3 Initialization

To obtain an instance of the [MSMQOutgoingQueueManagement](#) object:

- Clients MUST create an instance of the [MSMQManagement](#) object and initialize it by invoking the [IMSMQManagement::Init](#) method on the object. Refer to section [3.3.3](#) for the initialization of the MSMQManagement coclass.
- Clients MUST obtain the instance of MSMQOutgoingQueueManagement by invoking a call to [IUnknown::QueryInterface](#) (see section [3.1](#)) on the MSMQManagement object, with the interface identifier of [IMSMQOutgoingQueueManagement](#).
 - If the represented [Queue] of the MSMQManagement object is not an [Outgoing Queue], then the server MUST return E_NOINTERFACE (0x80004002).

3.5.4 Message Processing Events and Sequencing Rules

The [MSMQOutgoingQueueManagement](#) coclass defines a single interface: [IMSMQOutgoingQueueManagement](#).

3.5.4.1 IMSMQOutgoingQueueManagement Interface

The **IMSMQOutgoingQueueManagement** method provides methods that provide information on, and control of, an outgoing queue. **IMSMQOutgoingQueueManagement** inherits opnums 0 through 15 from the [IMSMQManagement \(section 3.3.4.1\)](#) interface. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {0188401c-247a-4fed-99c6-bf14119d7055} (coclass MSMQOutgoingQueueManagement as specified in section [1.9](#)), which implements the IMSMQOutgoingQueueManagement interface using the UUID {64c478fb-f9b0-4695-8a7f-439ac94326d3}.

Methods in RPC Opnum Order

Method	Description
State (section 3.5.4.1.1) , get State	Identifies the connection status of the outgoing queue represented by this object. Opnum: 16
NextHops (section 3.5.4.1.2) , get NextHops	Returns the address, or an array of possible addresses, for routing messages to the intermediate queue manager in the next hop. Opnum: 17
EodGetSendInfo (section 3.5.4.1.3)	Returns a collection of named Exactly Once Delivery (EOD) properties of the outgoing queue represented by this object. Opnum: 18
Resume (section 3.5.4.1.4)	Resumes the transfer of messages from the outgoing queue represented by this object. Opnum: 19
Pause (section 3.5.4.1.5)	Pauses the transfer of messages from the outgoing queue represented by this object.

Method	Description
	Opnum: 20
FodResend (section 3.5.4.1.6)	Resends the pending sequence of transactional messages in the outgoing queue represented by this object. Opnum: 21

3.5.4.1.1 State (Opnum 16)

The **State** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Outgoing Queue State] property of the represented [Outgoing Queue].

```
[propget] HRESULT State(
    [out, retval] long* plState
);
```

plState: A pointer to a long that corresponds to the [QUEUE_STATE \(section 2.2.2.19\)](#) enumeration.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Outgoing Queue] from the [Queues Table] of the *iQueueManager*, where the [Destination Format Name] of the [Outgoing Queue] matches the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Outgoing Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Outgoing Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.

- The *p/State* output variable MUST be set to the [Outgoing Queue State] property of the identified [Outgoing Queue].

3.5.4.1.2 NextHops (Opnum 17)

The **NextHops** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Next Hops] property of the represented [Outgoing Queue].

```
[propget] HRESULT NextHops(
    [out, retval] VARIANT* pvNextHops
);
```

pvNextHops: A pointer to a VARIANT that contains an array of zero or more strings (VT_ARRAY | VT_BSTR) that specify the routing addresses.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Outgoing Queue] from the [Queues Table] of the *iQueueManager*, where the [Destination Format Name] of the [Outgoing Queue] matches the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Outgoing Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Outgoing Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.
- The *pvNextHops* output variable MUST be set to the [Next Hops] property of the identified [Outgoing Queue].

3.5.4.1.3 EodGetSendInfo (Opnum 18)

The **EodGetSendInfo** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [EODSourceInfo] property of the represented [Outgoing Queue].

```
HRESULT EodGetSendInfo(  
    [out, retval] IMSMQCollection** ppCollection  
);
```

ppCollection: A pointer to an [EODSourceInfo \(section 2.2.3.2\)](#) collection.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Outgoing Queue] from the [Queues Table] of the *iQueueManager*, where the [Destination Format Name] of the [Outgoing Queue] matches the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Outgoing Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Outgoing Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE (0xC00E0004), and take no further action.
- The *ppCollection* output variable MUST be set to the [EODSourceInfo] property of the identified [Outgoing Queue].

3.5.4.1.4 Resume (Opnum 19)

The **Resume** method is received by the server in an RPC_REQUEST packet. In response, the server resumes the transfer of messages from the represented outgoing [Queue].

```
HRESULT Resume();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<25>](#)

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Outgoing Queue] from the [Queues Table] of the *iQueueManager*, where the [Destination Format Name] of the [Outgoing Queue] matches the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Outgoing Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Outgoing Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE.
- If the [Outgoing Queue State] of the [Outgoing Queue] is not MQ_QUEUE_STATE_ONHOLD (0x00000004):
 - The server MUST return S_OK and take no further action.
- The *iQueueManager* MUST attempt to resume the transfer of messages from the [Outgoing Queue]. For more details on the message transfer process, refer to section [1.4.1](#).
- The [Outgoing Queue State] of the identified [Outgoing Queue] MUST not be MQ_QUEUE_STATE_ONHOLD (0x00000004) upon successful completion of this call.

3.5.4.1.5 Pause (Opnum 20)

The **Pause** method is received by the server in an RPC_REQUEST packet. In response, the server pauses the transmission of messages from the referenced outgoing [Queue].

```
HRESULT Pause();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<26>](#)

When processing this call, the server MUST follow these guidelines:

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Outgoing Queue] from the [Queues Table] of the *iQueueManager*, where the [Destination Format Name] of the [Outgoing Queue] matches the *QueueFormatName* instance variable.
- The server MUST return an error HRESULT if no [Outgoing Queue] is found, and take no further action.
- If the [Outgoing Queue State] of the [Outgoing Queue] is MQ_QUEUE_STATE_ONHOLD (0x00000004):
 - The server MUST return S_OK and take no further action.
- The server MUST set the [Outgoing Queue State] property of the *iQueueManager* to MQ_QUEUE_STATE_ONHOLD (0x00000004).
- The *iQueueManager* MUST attempt to pause the transfer of messages from the [Outgoing Queue]. For more details on the message transfer process, refer to section [1.4.1](#).

3.5.4.1.6 EodResend (Opnum 21)

The **EodResend** method is received by the server in an RPC_REQUEST packet. In response, the server resends the pending sequence of transactional messages in the represented [Outgoing Queue].

```
HRESULT EodResend();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<27>](#)

- If the *ObjectIsInitialized* instance variable is False:
 - The server MUST return MQ_ERROR_UNINITIALIZED_OBJECT (0xC00E0094), and take no further action.
- The server MUST identify the [Queue Manager], referred to here as *iQueueManager*, as follows:
 - If the *ComputerName* instance variable is NULL:
 - Find the [Local Queue Manager].
 - Else:
 - Find the [Queue Manager] where the [Computer Name] matches the *ComputerName* instance variable.
 - The server MUST return an error HRESULT if no [Queue Manager] is found, and take no further action.
- Look up the [Outgoing Queue] from the [Queues Table] of the *iQueueManager*, where the [Destination Format Name] of the [Outgoing Queue] matches the *QueueFormatName* instance variable.
- The server MUST return with an error HRESULT if no [Outgoing Queue] is found, and take no further action.
- If the [IsActive] property of the identified [Outgoing Queue] is False:
 - The server MUST return MQ_ERROR_QUEUE_NOT_ACTIVE.
- The server MUST notify *iQueueManager* to resend the pending sequence of messages from the identified [Outgoing Queue].

3.5.5 Timer Events

No timer events are required.

3.5.6 Other Local Events

No local events are required.

3.6 MSMQTransactionDispenser Coclass

The MSMQTransactionDispenser object allows clients to start a new **internal transaction** and obtain an [MSMQTransaction](#) object that represents the underlying, newly created transaction. The MSMQTransaction object can then be used in calls to send or receive messages.

3.6.1 Abstract Data Model

The [MSMQTransactionDispenser](#) object does not maintain any state.

3.6.2 Timers

No protocol timers are required.

3.6.3 Initialization

No initialization is required for this coclass.

3.6.4 Message Processing Events and Sequencing Rules

This coclass includes three interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQTransactionDispenser3	IMSMQTransactionDispenser2	IMSMQTransactionDispenser
7	BeginTransaction	X	X	X
8	get Properties	X	X	

3.6.4.1 IMSMQTransactionDispenser3 Interface

The **IMSMQTransactionDispenser3** interface provides methods that enable transaction processing. The version number for this interface is 4.0.

There are two previous versions of this interface, **IMSMQTransactionDispenser** and **IMSMQTransactionDispenser2**. These previous versions are nearly identical but contain one less method. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server **MUST** implement a DCOM object class with the CLSID {d7d6e084-dccd-11d0-aa4b-0060970debae} (coclass **MSMQTransactionDispenser** as specified in section 1.9), which implements the **IMSMQTransactionDispenser3** interface using the UUID {eba96b15-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the **IUnknown** interface, as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the **IDispatch** interface, as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
BeginTransaction (section 3.6.4.1.1)	Initiates a new internal transaction and returns an IMSMQTransaction3 object that represents the underlying newly created transaction. <28> Opnum: 7
Properties (section 3.6.4.1.2) , get Properties	This method is not implemented. <29> Opnum: 8

3.6.4.1.1 BeginTransaction (Opnum 7)

The **BeginTransaction** method is received by the server in an RPC_REQUEST packet. In response, the server initiates a new internal transaction, and enlists the newly created transaction on the [Local Queue Manager]. This method returns an [IMSMQTransaction3](#) object that represents the underlying newly created transaction.

```
HRESULT BeginTransaction(  
    [out, retval] IMSMQTransaction3** ptransaction  
);
```

ptransaction: A pointer to an **IMSMQTransaction3** pointer that represents the newly created [Enlisted Transaction].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<30>](#)

When processing this call, the server MUST follow these guidelines:

- The server MUST create an internal transaction object that implements the [ITransaction \(section 3.8.4.1\)](#) interface.
- The server MUST set the *isInternal* instance variable of the created transaction object to True.
- The server MUST create a new [Enlisted Transaction] and set the [Transaction Identifier] property to the value of the *TransactionIdentifier* instance variable of the created transaction object.
- The server MUST add the created [Enlisted Transaction] to the [Enlisted Transactions Table] in the [Local Queue Manager].
- The server MUST create an **IMSMQTransaction3** object, and initialize it with the created transaction object.
- The *ptransaction* output variable MUST be set to the **IMSMQTransaction3** object.

3.6.4.1.2 Properties (Opnum 8)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(  
    [out, retval] IDispatch** ppcolProperties  
);
```

ppcolProperties: A pointer to an IDispatch pointer (see [\[MS-OAUT\]](#) section 3.1.4). The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The server MUST take no action and return E_NOTIMPL (0x80004001).

3.6.5 Timer Events

No timer events are required.

3.6.6 Other Local Events

No local events are required.

3.7 MSMQCoordinatedTransactionDispenser Coclass

The MSMQCoordinatedTransactionDispenser object allows clients to start a new Microsoft Distributed Transaction Coordinator (MS DTC) transaction and obtain an [MSMQTransaction](#) object that represents the underlying newly created transaction. The MSMQTransaction object can then be used in calls to send or receive messages.

3.7.1 Abstract Data Model

The [MSMQCoordinatedTransactionDispenser](#) object does not maintain any state.

3.7.2 Timers

No protocol timers are required.

3.7.3 Initialization

No initialization is required.

3.7.4 Message Processing Events and Sequencing Rules

This coclass includes three interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opn um	Method name (in the most recent interface revision)	IMSMQCoordinatedTransactionDispenser3	IMSMQCoordinatedTransactionDispenser2	IMSMQCoordinatedTransactionDispenser
7	BeginTransaction	X	X	X
8	getProperties	X	X	

3.7.4.1 IMSMQCoordinatedTransactionDispenser3 Interface

The **IMSMQCoordinatedTransactionDispenser3** interface provides methods that enable transaction processing. The version number for this interface is 4.0.

There are two previous versions of this interface, IMSMQCoordinatedTransactionDispenser and IMSMQCoordinatedTransactionDispenser2. These previous versions are nearly identical but have one less method. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e082-dccd-11d0-aa4b-0060970debae} (coclass

MSMQCoordinatedTransactionDispenser as specified in section [1.9](#)), which implements the IMSMQCoordinatedTransactionDispenser3 interface using the UUID {eba96b14-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface, as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**. Opnums 3 through 6 are inherited from the IDispatch interface, as specified in [\[MS-OAUT\]](#) section **3.1.4**.

Methods in RPC Opnum Order

Method	Description
BeginTransaction (section 3.7.4.1.1)	Initiates a new MS DTC transaction, and returns an IMSMQTransaction3 object that represents the underlying newly created transaction. <31> Opnum: 7
Properties (section 3.7.4.1.2), get Properties	This method is not implemented. <32> Opnum: 8

3.7.4.1.1 BeginTransaction (Opnum 7)

The **BeginTransaction** method is received by the server in an RPC_REQUEST packet. In response, the server initiates a new MS DTC transaction. This method returns an [IMSMQTransaction3](#) object that represents the underlying newly created transaction.

```
HRESULT BeginTransaction(  
    [out, retval] IMSMQTransaction3** ptransaction  
);
```

ptransaction: A pointer to an **IMSMQTransaction3** pointer that represents the newly created transaction.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.[<33>](#)

When processing this call, the server MUST follow these guidelines:

- The server MUST retrieve a **handle** to the MS DTC transaction manager by invoking the MS DTC-specific API.[<34>](#)
- The server MUST initiate a new distributed transaction using the MS DTC transaction manager. For more details on beginning a new distributed transaction, refer to [\[MS-DTCO\]](#) section 2.2.8.1.1.2.
- The server MUST retrieve the transaction object that represents the newly created distributed transaction, and implement the [ITransaction \(section 3.8.4.1\)](#) interface from the MS DTC transaction manager.[<35>](#)
- The server MUST create an **IMSMQTransaction3** object and initialize it with the transaction object.
- The *ptransaction* output variable MUST be set to the **IMSMQTransaction3** object.

3.7.4.1.2 Properties (Opnum 8)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(  
    [out, retval] IDispatch** ppcolProperties  
);
```

ppcolProperties: A pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The server MUST take no action and return E_NOTIMPL (0x80004001).

3.7.5 Timer Events

No timer events are required.

3.7.6 Other Local Events

No local events are required.

3.8 ITransaction Implementation Class

The ITransaction implementation class represents a transaction object. This transaction object is uniquely identified by using a transaction identifier, and can be used to commit, abort, and obtain status information about transactions.

3.8.1 Abstract Data Model

The [ITransaction](#) implementation maintains the following data elements:

- *TransactionIdentifier*: A GUID that uniquely identifies the transaction object.
- *IsInternal*: A Boolean flag that indicates whether the transaction is an internal transaction or an external MS DTC transaction.
- *isCommittedorAborted*: A Boolean flag that indicates whether the transaction has been committed or aborted.

3.8.2 Timers

No protocol timers are required.

3.8.3 Initialization

- The server MUST create a new GUID as specified in [RFC4122](#), and assign it to the *TransactionIdentifier* instance variable.
- The server MUST initialize the *isInternal* instance variable to False and the *isCommittedorAborted* instance variable to False.

3.8.4 Message Processing Events and Sequencing Rules

The [ITransaction](#) implementation class defines a single interface: [ITransaction](#).

3.8.4.1 ITransaction Interface

The **ITransaction** interface provides methods that complete transaction processing. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server **MUST** implement a DCOM interface using the UUID {0fb15084-af41-11ce-bd2b-204c4f4f5020}.

The opnum table below begins at opnum 3. Opnums 0 through 2 are inherited from the IUnknown interface, as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**.

Methods in RPC Opnum Order

Method	Description
Commit (section 3.8.4.1.1)	Commits the transaction. Opnum: 3
Abort (section 3.8.4.1.2)	Aborts the transaction. Opnum: 4
GetTransactionInfo (section 3.8.4.1.3)	Retrieves information about the transaction. Opnum: 5

3.8.4.1.1 Commit (Opnum 3)

The **Commit** method is received by the server in an RPC_REQUEST packet. In response, the server commits the transaction that it represents.

```
HRESULT Commit(  
    [in] short fRetaining,  
    [in] DWORD grfTC,  
    [in] DWORD grfRM  
);
```

fRetaining: A short that represents a Boolean value, that specifies whether or not to retain the transaction when it is finished.

grfTC: A DWORD that corresponds to the [XACTTC \(section 2.2.2.21\)](#) enumeration.

grfRM: A DWORD that is reserved for future use. For more details on this input parameter, refer to [\[MS-DTCO\]](#) section 2.2.7.1.

Return Values: The method **MUST** return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<36>](#)

When processing this call, the server **MUST** follow these guidelines:

- If the *isInternal* instance variable is True:

- If the *grfRM* input parameter is NOT equal to 0, or if the *fRetaining* input parameter is equal to True, or if the *grfTC* input parameter is not equal to XACTTC_SYNC (0x00000002):
 - The server MUST return XACT_E_NOTSUPPORTED (0x8004D00F), and take no further action.
- If the *isCommittedorAborted* instance variable is True:
 - The server MUST return MQ_ERROR_TRANSACTION_SEQUENCE (0xC00E0051), and take no further action.
- The server MUST fire a [Transaction Commit] event to notify the [Local Queue Manager] to commit the transaction. The [Transaction Commit] event will carry the *TransactionIdentifier* instance variable of this transaction object. Refer to section [3.1.6.1](#) for details on the outcome of this event.
- Else:
 - The server MUST send a commit request to the MS DTC transaction manager, which eventually triggers the [Transaction Commit] event on the [Queue Manager]s that are enlisted in the transaction. The [Transaction Commit] event will carry the *TransactionIdentifier* instance variable of this transaction object. For more details on the commit request to the MS DTC transaction manager, refer to [\[MS-DTCO\]](#) section 2.2.8.1.2.3. For more details on the outcome of the [Transaction Commit] event, refer to section [3.1.6.1](#).
 - If any errors are generated by the MS DTC transaction manager:
 - The server MUST return an error HRESULT and take no further action.
- The server MUST set the *isCommittedorAborted* instance variable to True.

3.8.4.1.2 Abort (Opnum 4)

The **Abort** method is received by the server in an RPC_REQUEST packet. In response, the server aborts the transaction that it represents.

```
HRESULT Abort(
    [in, unique] BOID* pboidReason,
    [in] short fRetaining,
    [in] short fAsync
);
```

pboidReason: A pointer to a BOID that indicates the reason for aborting the transaction. A BOID structure is defined as follows:

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

The array of bytes describes the reason for aborting the transaction.

fRetaining: A short that represents a Boolean value, that specifies whether or not to retain the transaction when it is finished.

fAsync: A short that represents a Boolean value, that specifies whether the abort is done synchronously (False) or asynchronously (True).

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<37>](#)

When processing this call, the server MUST follow these guidelines:

- If the *isInternal* instance variable is True:
 - If the *fRetaining* input parameter is equal to True, or if the *fAsync* input parameter is equal to True:
 - The server MUST return XACT_E_NOTSUPPORTED (0x8004D00F), and take no further action.
 - If the *isCommittedorAborted* instance variable is True:
 - The server MUST return MQ_ERROR_TRANSACTION_SEQUENCE (0xC00E0051), and take no further action.
 - The server MUST fire an [Transaction Abort] event to notify the [Local Queue Manager] to abort the transaction. The [Transaction Abort] event will carry the *TransactionIdentifier* instance variable of this transaction object. Refer to section [3.1.6.2](#) for details on the outcome of this event.
- Else:
 - The server MUST send an abort request to the MS DTC transaction manager, which eventually triggers the [Transaction Abort] event on the [Queue Manager]s that are enlisted in the transaction. The [Transaction Abort] event will carry the *TransactionIdentifier* instance variable of this transaction object. For more details on the abort request to the MS DTC transaction manager, refer to [\[MS-DTCO\]](#) section 2.2.8.1.2.1. For more details on the outcome of the [Transaction Abort] event, refer to section [3.1.6.2](#).
 - If any errors are generated by the MS DTC transaction manager:
 - The server MUST return an error HRESULT and take no further action.
- The server MUST set the *isCommittedorAborted* instance variable to True.

3.8.4.1.3 GetTransactionInfo (Opnum 5)

The **GetTransactionInfo** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves information about the transaction.

```
HRESULT GetTransactionInfo(  
    [out] XACTTRANSINFO* pinfo  
);
```

pinfo: A pointer to the caller-allocated XACTTRANSINFO structure in which the method returns information about the transaction.

The XACTTRANSINFO structure is defined as follows:

```
typedef struct XACTTRANSINFO {
```

```

    XACTUOW uow;
    ISOLEVEL isoLevel;
    ULONG isoFlags;
    DWORD grfTCSupported;
    DWORD grfRMSupported;
    DWORD grfTCSupportedRetaining;
    DWORD grfRMSupportedRetaining;
} XACTTRANSINFO;

```

Where:

- The XACTUOW structure is defined in [\[MS-MQMP\]](#) section **2.2.2.1**.
- The ISOLEVEL type corresponds to the OLETX_ISOLATION_LEVEL enumeration, as defined in [\[MS-DTCO\]](#) section **2.2.6.9**.
- The isoFlags property corresponds to the OLETX_ISOLATION_FLAGS enumeration, as defined in [\[MS-DTCO\]](#) section **2.2.6.8**.
- The grfTCSupported property specifies a bitmask that indicates which [XACTTC \(section 2.2.2.21\)](#) flags this transaction implementation supports.
- The grfRMSupported, grfTCSupportedRetaining, and grfRMSupportedRetaining properties are reserved for future use, and the server MUST set a value of 0 for these properties.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<38>](#)

When processing this call, the server MUST follow these guidelines:

- If the *pinfo* output variable is NULL:
 - The server MUST return an error HRESULT and take no further action.
- If the *isInternal* instance variable is True:
 - The server MUST set the uow property of the *pinfo* output variable to the value of the *TransactionIdentifier* instance variable.
 - The server MUST set all the remaining properties of *pinfo* output variable to 0.
- Else:
 - The server MUST set the uow property of the *pinfo* output variable to the value of the *TransactionIdentifier* instance variable.
 - The server MUST send a request to the MS DTC transaction manager to retrieve the transaction details using the *TransactionIdentifier* instance variable. For more details on this request, refer to [\[MS-DTCO\]](#) section 2.2.8.3.1.1.
 - The server MUST set the remaining properties of the *pinfo* output variable to the values obtained from the above request.

3.8.5 Timer Events

No timer events are required.

3.8.6 Other Local Events

No local events are required.

3.9 MSMQTransaction Coclass

The MSMQTransaction object identifies an underlying transaction object that is obtained externally using an [MSMQCoordinatedTransactionDispenser](#) object, or created internally using an [MSMQTransactionDispenser](#) object, or by attaching to an existing transaction. The MSMQTransaction object can be used in the following ways:

- To commit or abort the transaction.
- To retrieve the underlying transaction object.
- As the *Transaction* argument to one of the following methods:
 - When sending messages:
 - [IMSMQMessage::Send](#)
 - When retrieving messages:
 - [IMSMQQueue4::Receive](#)
 - [IMSMQQueue4::ReceiveCurrent](#)
 - [IMSMQQueue4::ReceiveFirstByLookupId](#)
 - [IMSMQQueue4::ReceivePreviousByLookupId](#)
 - [IMSMQQueue4::ReceiveByLookupId](#)
 - [IMSMQQueue4::ReceiveNextByLookupId](#)
 - [IMSMQQueue4::ReceiveLastByLookupId](#)

3.9.1 Abstract Data Model

An implementation of the [MSMQTransaction](#) coclass maintains a reference to the underlying transaction object. This transaction reference can be used to commit or abort the transaction.

The MSMQTransaction coclass maintains the following instance data element:

- *Transaction*: This instance variable holds a reference to the transaction object that is implementing the [ITransaction](#) interface.

3.9.2 Timers

No protocol timers are required.

3.9.3 Initialization

The server MUST initialize the *Transaction* instance variable to NULL.

3.9.4 Message Processing Events and Sequencing Rules

This coclass includes three interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQTransaction3	IMSMQTransaction2	IMSMQTransaction
7	get Transaction	X	X	X
8	Commit	X	X	X
9	Abort	X	X	X
10	InitNew	X	X	
11	get Properties	X	X	
12	get ITransaction	X		

3.9.4.1 IMSMQTransaction Interface

The **IMSMQTransaction** interface provides methods that enable transaction processing. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e080-dccd-11d0-aa4b-0060970debae} (coclass MSMQTransaction as specified in section 1.9), which implements the IMSMQTransaction interface using the UUID {d7d6e07f-dccd-11d0-aa4b-0060970debae}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface, as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the IDispatch interface, as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
Transaction (section 3.9.4.1.1) , get Transaction	This method is deprecated and should not be implemented. Opnum: 7
Commit (section 3.9.4.1.2)	Commits the underlying internal transaction or MS DTC transaction. Opnum: 8
Abort (section 3.9.4.1.3)	Aborts the underlying internal transaction or MS DTC transaction. Opnum: 9

3.9.4.1.1 Transaction (Opnum 7)

The **Transaction** method is received by the server in an RPC_REQUEST packet. In response, the server returns a long variable that represents the underlying transaction object.

```
[propget] HRESULT Transaction(  
    [out, retval] long* plTransaction  
);
```

plTransaction: A pointer to a long that identifies the underlying transaction.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

This method is deprecated, and the server SHOULD NOT implement it. [<39>](#) The server SHOULD return E_NOTIMPL (0x80004001).

3.9.4.1.2 Commit (Opnum 8)

The **Commit** method is received by the server in an RPC_REQUEST packet. In response, the server commits the underlying internal transaction or MS DTC external transaction.

```
HRESULT Commit(  
    [in, optional] VARIANT* fRetaining,  
    [in, optional] VARIANT* grfTC,  
    [in, optional] VARIANT* grfRM  
);
```

fRetaining: A pointer to a VARIANT containing a VARIANT_BOOL. This flag specifies whether or not to retain the transaction when it is finished.

grfTC: A pointer to a VARIANT containing a VT_I4 integer that corresponds to the [XACTTC \(section 2.2.2.21\)](#) enumeration.

If not specified by the client, the server MUST use the default value of XACTTC_SYNC (0x00000002) in place of the unspecified value.

grfRM: A pointer to a VARIANT that contains a long. For more details on this input parameter, refer to [\[MS-DTCO\]](#) section **2.2.7.1**. If not specified by the client, the server MUST use the default value of 0 in place of the unspecified value.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *Transaction* instance variable is NULL:
 - The server MUST return E_INVALIDARG (0x80070057), and take no further action.
- The server MUST commit the transaction by invoking the Commit method ([ITransaction::Commit \(section 3.8.4.1.1\)](#)) on the *Transaction* instance variable, passing the input parameters *fRetaining*, *grfTC*, and *grfRM*.

3.9.4.1.3 Abort (Opnum 9)

The **Abort** method is received by the server in an RPC_REQUEST packet. In response, the server aborts the message queuing internal transaction or MS DTC external transaction.

```
HRESULT Abort(  
    [in, optional] VARIANT* fRetaining,  
    [in, optional] VARIANT* fAsync  
);
```

fRetaining: A pointer to a VARIANT containing a VARIANT_BOOL. This flag specifies whether or not to retain the transaction when it is finished.

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

fAsync: A pointer to a VARIANT containing a VARIANT_BOOL that specifies whether the abort is done synchronously (False) or asynchronously (True). If not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *Transaction* instance variable is NULL:
 - The server MUST return E_INVALIDARG (0x80070057), and take no further action.
- The server MUST abort the transaction by invoking the Abort method ([ITransaction::Abort \(section 3.8.4.1.2\)](#)) on the *Transaction* instance variable, passing the input parameters *fRetaining* and *fAsync*.

3.9.4.2 IMSMQTransaction2 Interface

The **IMSMQTransaction2** interface provides methods that return information about the queue manager on a given server. **IMSMQTransaction2** (section 3.9.4.2) inherits opnums 0 through 9 from the [IMSMQTransaction](#) interface. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e080-dccd-11d0-aa4b-0060970debae} (coclass MSMQTransaction as specified in section 1.9), which implements the IMSMQTransaction2 interface using the UUID {2ce0c5b0-6e67-11d2-b0e6-00e02c074f6b}.

Methods in RPC Opnum Order

Method	Description
InitNew (section 3.9.4.2.1)	Initializes the object to represent an existing underlying transaction object. Opnum: 10
Properties (section 3.9.4.2.2) , get Properties	This method is not implemented. Opnum: 11

3.9.4.2.1 InitNew (Opnum 10)

The **InitNew** method is received by the server in an RPC_REQUEST packet. In response, the server initializes an MSMQ transaction object to represent an existing underlying transaction object.

```
HRESULT InitNew(  
    [in] VARIANT varTransaction  
);
```

varTransaction: A pointer to a VARIANT that points to an existing underlying transaction object that is implementing the [ITransaction](#) interface. The VARIANT that is passed could be any one of the following types:

- VT_UNKNOWN
- VT_UNKNOWN | VT_BYREF
- VT_DISPATCH
- VT_DISPATCH | VT_BYREF
- VT_INTPTR
- VT_INTPTR | VT_BYREF

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *Transaction* instance variable is NOT NULL:
 - The server MUST return MQ_ERROR_TRANSACTION_USAGE (0xC00E0050), and take no further action.
- The server MUST retrieve the transaction object that is implementing the **ITransaction** interface by invoking [IUnknown::QueryInterface](#) (refer to section 3.1) on the *varTransaction* input parameter, passing the interface identifier of **ITransaction**.
- The server MUST return E_INVALIDARG (0x80070057) if the *varTransaction* input parameter does not implement the **ITransaction** interface, and take no further action.
- The server MUST set the *Transaction* instance variable to the value of the transaction object obtained above.

3.9.4.2.2 Properties (Opnum 11)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(  
    [out, retval] IDispatch** ppcolProperties  
);
```

ppcolProperties: A pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The server MUST take no action and return E_NOTIMPL (0x80004001).

3.9.4.3 IMSMQTransaction3 Interface

The **IMSMQTransaction3** interface provides methods that return information about the queue manager on a given server. **IMSMQTransaction3** inherits opnums 0 through 11 from the [IMSMQTransaction2 \(section 3.9.4.2\)](#) interface. The version number for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e080-dccd-11d0-aa4b-0060970debae} (coclass MSMQTransaction as specified in section 1.9), which implements the IMSMQTransaction3 interface using the UUID {eba96b13-2168-11d3-898c-00e02c074f6b}.

Methods in RPC Opnum Order

Method	Description
ITransaction (section 3.9.4.3.1) , get ITransactcion	Returns the ITransaction interface on the underlying transaction object. Opnum: 12

3.9.4.3.1 ITransaction (Opnum 12)

The **ITransaction** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [ITransaction](#) interface on the underlying transaction object.

```
[propget] HRESULT ITransaction(  
    [out, retval] VARIANT* pvarITransaction  
);
```

pvarITransaction: A pointer to a VARIANT (VT_UNKNOWN or VT_EMPTY), that upon successful completion contains the underlying transaction object.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

The *pvarITransaction* output variable MUST be set to the *Transaction* instance variable.

3.9.5 Timer Events

No timer events are required.

3.9.6 Other Local Events

No local events are required.

3.10 MSMQQueueInfo Coclass

The MSMQQueueInfo object represents a [Queue], referred to here as *referenced queue*.

The MSMQQueueInfo object can be used to do the following operations:

- Create an [Application Queue].
- Open an existing [Queue].
- Delete an existing [Application Queue].
- Get and set [Application Queue] properties.

3.10.1 Abstract Data Model

An implementation of the [MSMQQueueInfo](#) coclass maintains the following abstract data elements:

- *QueueFormatName*: A string that contains the format name that identifies the *referenced queue*.
- *QueuePathName*: A string that contains the path name that identifies the *referenced queue*.
- *IdentifierGUID*: A GUID that uniquely identifies the *referenced queue*. This value is set when the *referenced queue* is created, or when the client invokes the [MSMQQueueInfo.Refresh](#) method.

This instance variable is read-only, and maps to the *referenced queue*'s [Identifier GUID] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- *ServiceTypeGUID*: An application-specific GUID that classifies the *referenced queue*.

This instance variable maps to the *referenced queue*'s [Service Type GUID] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- *Label*: A text string that MAY contain a client-defined description for the *referenced queue*.

This instance variable maps to the *referenced queue*'s [Label] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- *IsTransactional*: A Boolean flag that, when True, indicates that the *referenced queue* is a transactional queue.

This value is set when the *referenced queue* is created, or when the client invokes the **MSMQQueueInfo.Refresh** method.

This instance variable is read-only and maps to the *referenced queue*'s [IsTransactional] property.

- *IsWorldReadable*: A Boolean that is True when the *referenced queue* is accessible to any user.

Otherwise, if False, the *referenced queue* is only accessible to the [Owner] of the *referenced queue*, and to the Administrators.

This instance variable maps to the *referenced queue*'s [IsWorldReadable] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- *PrivacyLevel*: An enumeration that indicates whether the *referenced queue* restricts [Message]s according to their [Privacy Level] value. The enumeration values are defined in [MQPRIVLEVEL \(section 2.2.2.7\)](#).

This instance variable maps to the *referenced queue*'s [Privacy Level] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- **IsTargetJournalingEnabled:** A Boolean flag that, when True, indicates that messages received from the [Messages Table] of the *referenced queue* are recorded in the *referenced queue*'s [Journal Queue].[Messages Table].

This instance variable maps to the *referenced queue*'s [IsTargetJournalingEnabled] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- **Quota:** A number indicating the maximum size of the [Messages Table] of the *referenced queue*, expressed in kilobytes.

This instance variable maps to the *referenced queue*'s [Quota] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- **BasePriority:** A numeric value between -32768 and 32767. Implementations of the message transfer process MAY interpret [Base Priority] relative to other [Application Queue]s to prioritize message transfer work. A greater value indicates higher priority.

This instance variable maps to the *referenced queue*'s [Base Priority] property only if the *referenced queue* is a [Public Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- **CreationTime:** A date/time value that indicates when the *referenced queue* was created.

This instance variable is read-only, and maps to the *referenced queue*'s [Creation Time] property.

- **ModificationTime:** A date/time value that indicates when any element of the *referenced queue* was most recently modified.

This instance variable is a read-only, and maps to the *referenced queue*'s [Modification Time] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- **AuthenticationRequired:** A Boolean flag that, when True, indicates that the *referenced queue* accepts only [Message]s in which the [Authentication Level] indicates that the [Message] was signed. When False, the *referenced queue* does not mandate that messages must be signed.

This instance variable maps to the *referenced queue*'s [Authentication Required] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- **JournalQuota:** A number specifying the maximum size, in kilobytes, that is allowed for the [Messages Table] of the [Journal Queue] that is referenced by the [Journal Queue] property of the *referenced queue*.

This instance variable maps to the [Quota] property of the [Journal Queue] that is referenced by the [Journal Queue] property of the *referenced queue* only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

- **MulticastAddress:** A string that contains an IP multicast address on which the respective [Queue Manager] will receive [Message]s for this *reference queue*.

This instance variable maps to the *referenced queue's* [Multicast Address] property only if the *referenced queue* is an [Application Queue]. Otherwise, the value of this instance variable remains as described in the initialization section [3.10.3](#).

3.10.2 Timers

No protocol timers are required.

3.10.3 Initialization

The [MSMQQueueInfo](#) class can be constructed and initialized by either the client or by invoking [IMSMQQuery4::LookupQueue](#) or [IMSMQQuery4::LookupQueue_v2](#).

- When **IMSMQQuery4::LookupQueue** or **IMSMQQuery4::LookupQueue_v2** is invoked, the [MSMQQuery](#) object returns an [MSMQQueueInfos](#) collection that initializes the MSMQQueueInfo object instances on-demand when the client iterates over the collection.
- When the client constructs the object, the instance variables will be set with the following default values:

Instance variable	Default value
QueuePathName QueueFormatName IdentifierGUID ServiceTypeGUID	NULL
Label	Empty string
IsTransactional	FALSE (0x00000000)
IsWorldReadable	FALSE (0x00000000)
PrivacyLevel	MQ_PRIV_LEVEL_OPTIONAL (0x00000001)
IsTargetJournalingEnabled	FALSE (0x00000000)
Quota	0xffffffff
BasePriority	0x00000003
AuthenticationLevel	FALSE (0x00000000)
JournalQuota	0xffffffff
MulticastAddress	Empty string
ModificationTime	time_t zero (0x00000000), equivalent to Variant DateTime Midnight, January 1st, 1970 UTC.

Instance variable	Default value
AuthenticationRequired	FALSE (0x00000000)

- If the client constructed the MSMQQueueInfo object to create an [Application Queue]:
 - The client MUST call [put IMSMQQueueInfo4::PathName](#) before calling [IMSMQQueueInfo4::Create](#).
- If the client constructed the MSMQQueueInfo object to open, refresh/update or delete a [Queue]:
 - The client MUST call [put IMSMQQueueInfo4::PathName](#) or [put IMSMQQueueInfo4::FormatName](#) before calling [IMSMQQueueInfo4::Open](#), [IMSMQQueueInfo4::Refresh](#), [IMSMQQueueInfo4::Update](#), or [IMSMQQueueInfo4::Delete](#).

3.10.4 Message Processing Events and Sequencing Rules

This coclass includes four interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQQueueInfo4	IMSMQQueueInfo3	IMSMQQueueInfo2	IMSMQQueueInfo
7	get QueueGuid	X	X	X	X
8	get ServiceTypeGuid	X	X	X	X
9	put ServiceTypeGuid	X	X	X	X
10	get Label	X	X	X	X
11	put Label	X	X	X	X
12	get PathName	X	X	X	X
13	put PathName	X	X	X	X
14	get FormatName	X	X	X	X
15	put FormatName	X	X	X	X
16	get IsTransactional	X	X	X	X

Opnum	Method name (in the most recent interface revision)	IMSMQueueInfo4	IMSMQueueInfo3	IMSMQueueInfo2	IMSMQueueInfo
17	get PrivLevel	X	X	X	X
18	put PrivLevel	X	X	X	X
19	get Journal	X	X	X	X
20	put Journal	X	X	X	X
21	get Quota	X	X	X	X
22	put Quota	X	X	X	X
23	get BasePriority	X	X	X	X
24	put BasePriority	X	X	X	X
25	get CreateTime	X	X	X	X
26	get ModifyTime	X	X	X	X
27	get Authenticate	X	X	X	X
28	put Authenticate	X	X	X	X
29	get JournalQuota	X	X	X	X
30	put JournalQuota	X	X	X	X
31	get IsWorldReadable	X	X	X	X
32	Create	X	X	X	X
33	Delete	X	X	X	X
34	Open	X	X	X	X
35	Refresh	X	X	X	X
36	Update	X	X	X	X
37	get PathNameDNS	X	X	X	

Opnum	Method name (in the most recent interface revision)	IMSMQQueueInfo4	IMSMQQueueInfo3	IMSMQQueueInfo2	IMSMQQueueInfo
38	get Properties	X	X	X	
39	get Security	X	X	X	
40	put Security	X	X	X	
41	get IsTransactional2	X	X		
42	get IsWorldReadable2	X	X		
43	get MulticastAddress	X	X		
44	put MulticastAddress	X	X		
45	get ADsPath	X	X		

3.10.4.1 MSMQQueueInfo4 Interface

The **IMSMQQueueInfo4** interface provides methods that return information about a queue on a given server. The version number for this interface is 4.0.

There are three previous versions of this interface: **IMSMQQueueInfo**, **IMSMQQueueInfo2**, and **IMSMQQueueInfo3**. These previous versions are nearly identical, but have a few less methods. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e07c-dccd-11d0-aa4b-0060970debae} (coclass **MSMQQueueInfo** as specified in section 1.9), which implements the **IMSMQQueueInfo4** interface using the UUID {d7d6e07c-dccd-11d0-aa4b-0060970debae}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the **IUnknown** interface, as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the **IDispatch** interface, as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
QueueGuid (section 3.10.4.1.1) , get QueueGuid	Returns the unique identifier of the queue. Opnum: 7
ServiceTypeGuid (section 3.10.4.1.2) , get ServiceTypeGuid	Returns an identifier indicating the type of service provided by the queue. Opnum: 8
ServiceTypeGuid (section 3.10.4.1.3) , put ServiceTypeGuid	Sets an identifier indicating the type of service provided by the queue. Opnum: 9
Label (section 3.10.4.1.4) , get Label	Returns the label of the queue. Opnum: 10
Label (section 3.10.4.1.5) , put Label	Sets the label of the queue. Opnum: 11
PathName (section 3.10.4.1.6) , get PathName	Returns the path name of the queue. Opnum: 12
PathName (section 3.10.4.1.7) , put PathName	Sets the path name of the queue. Opnum: 13
FormatName (section 3.10.4.1.8) , get FormatName	Returns the format name that was set when the queue was created. Opnum: 14
put FormatName (section 3.10.4.1.9) , put FormatName	Sets the format name used to identify the queue. Opnum: 15
IsTransactional (section 3.10.4.1.10) , get IsTransactional	Returns a value indicating whether the queue is a transactional queue. Opnum: 16
PrivLevel (section 3.10.4.1.11) , get PrivLevel	Returns the privacy level for the queue. Opnum: 17
PrivLevel (section 3.10.4.1.12) , put PrivLevel	Sets the privacy level for the queue. Opnum: 18
Journal (section 3.10.4.1.13) , get Journal	Returns the journaling level for the queue. Opnum: 19
Journal (section 3.10.4.1.14) , put Journal	Sets the journaling level the queue. Opnum: 20
Quota (section 3.10.4.1.15) , get Quota	Returns the maximum size (in kilobytes) of the queue. Opnum: 21
Quota (section 3.10.4.1.16) , put Quota	Sets the maximum size (in kilobytes) of the queue. Opnum: 22
BasePriority (section	Returns the base priority for all messages sent to the public

Method	Description
3.10.4.1.17) , get BasePriority	queue. Opnum: 23
BasePriority (section 3.10.4.1.18) , put BasePriority	Sets the base priority for all messages sent to the public queue. Opnum: 24
CreateTime (section 3.10.4.1.19) , get CreateTime	Returns a value indicating the date and time when the queue was created. Opnum: 25
ModifyTime (section 3.10.4.1.20) , get ModifyTime	Returns a value indicating the date and time when the queue's properties were last modified. Opnum: 26
Authenticate (section 3.10.4.1.21) , get Authenticate	Returns the authentication level for the queue. Opnum: 27
Authenticate (section 3.10.4.1.22) , put Authenticate	Sets the authentication level for the queue. Opnum: 28
JournalQuota (section 3.10.4.1.23) , get JournalQuota	Returns the maximum size (in kilobytes) of the queue journal. Opnum: 29
JournalQuota (section 3.10.4.1.24) , put JournalQuota	Sets the maximum size (in kilobytes) of the queue journal. Opnum: 30
IsWorldReadable (section 3.10.4.1.25) , get IsWorldReadable	Returns a value indicating whether everyone can read messages in the queue, or only the owner and administrators of the queue can read messages in it. Opnum: 31
Create (section 3.10.4.1.26)	Creates a new queue using the path name. Opnum: 32
Delete (section 3.10.4.1.27)	Deletes the queue using the format name or the path name. Opnum: 33
Open (section 3.10.4.1.28)	Opens the queue identified by the format name. <40> Opnum: 34
Refresh (section 3.10.4.1.29)	Refreshes the properties of the object with the values stored in the Directory Service (for public queues) or provided by the local queue manager (for private queues). Opnum: 35
Update (section 3.10.4.1.30)	Updates the properties stored in the Directory Service (for public queues) or the local queue manager (for private queues) with values from this object. Opnum: 36
PathNameDNS (section 3.10.4.1.31) , get PathNameDNS	Returns the DNS path name of the queue. <41> Opnum: 37

Method	Description
Properties (section 3.10.4.1.32) , get Properties	The method is not implemented. <42> Opnum: 38
Security (section 3.10.4.1.33) , get Security	The method is not implemented. <43> Opnum: 39
Security (section 3.10.4.1.34) , put Security	The method is not implemented. <44> Opnum: 40
IsTransactional2 (section 3.10.4.1.35) , get IsTransactional2	Returns a value indicating whether the queue is a transactional queue. <45> Opnum: 41
IsWorldReadable2 (section 3.10.4.1.36) , get IsWorldReadable2	Returns a value indicating whether everyone can read messages in the queue, or only the owner and administrators of the queue can read messages in it. <46> Opnum: 42
MulticastAddress (section 3.10.4.1.37) , get MulticastAddress	Returns the multicast addresses that is associated with the queue. <47> Opnum: 43
MulticastAddress (section 3.10.4.1.38) , put MulticastAddress	Sets the multicast addresses that is associated with the queue. <48> Opnum: 44
ADsPath (section 3.10.4.1.39) , get ADsPath	Returns the directory path to a public queue. <49> Opnum: 45

3.10.4.1.1 QueueGuid (Opnum 7)

The **QueueGuid** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *IdentifierGUID* instance variable that uniquely identifies the *referenced queue*.

```
[propget] HRESULT QueueGuid(
    [out, retval] BSTR* pbstrGuidQueue
);
```

pbstrGuidQueue: A pointer to a BSTR that represents a GUID.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pbstrGuidQueue* output parameter to the value of the *IdentifierGUID* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.2 ServiceTypeGuid (Opnum 8)

The **ServiceTypeGuid** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *ServiceTypeGuid* instance variable, which indicates the type of service provided by the *referenced queue*.

```
[propget] HRESULT ServiceTypeGuid(  
    [out, retval] BSTR* pbstrGuidServiceType  
);
```

pbstrGuidServiceType: A pointer to BSTR that represents a GUID.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pbstrGuidServiceType* output parameter to the value of the *ServiceTypeGuid* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.3 ServiceTypeGuid (Opnum 9)

The **ServiceTypeGuid** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *ServiceTypeGuid* instance variable, which indicates the type of service provided by the *referenced queue*.

```
[propput] HRESULT ServiceTypeGuid(  
    [in] BSTR bstrGuidServiceType  
);
```

bstrGuidServiceType: A BSTR that represents a GUID.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *ServiceTypeGuid* instance variable to the value of the *bstrGuidServiceType* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.4 Label (Opnum 10)

The **Label** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *Label* instance variable, which specifies the label of the *referenced queue*.

```
[propget] HRESULT Label(  
    [out, retval] BSTR* pbstrLabel  
);
```

pbstrLabel: A pointer to a BSTR that specifies the queue label.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pbstrLabel* output parameter to the value of the *Label* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.5 Label (Opnum 11)

The **Label** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *Label* instance variable, which specifies the label of the *referenced queue*.

```
[propput] HRESULT Label(  
    [in] BSTR bstrLabel  
);
```

bstrLabel: A BSTR that specifies the queue label.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *Label* instance variable to the value of the *bstrLabel* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.6 PathName (Opnum 12)

The **PathName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *QueuePathName* instance variable, which contains the path name of the *referenced queue*.

```
[propget] HRESULT PathName(  
    [out, retval] BSTR* pbstrPathName  
);
```

pbstrPathName: A pointer to BSTR that specifies the path name of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pbstrPathName* output parameter to the value of the *QueuePathName* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.7 PathName (Opnum 13)

The **PathName** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *QueuePathName* instance variable, which contains the path name of the *referenced queue*.

```
[propput] HRESULT PathName(  
    [in] BSTR bstrPathName  
);
```

bstrPathName: A BSTR that specifies the path name of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *QueuePathName* instance variable to the value of the *bstrPathName* input parameter.
- Set the *QueueFormatName* instance variable to the format name that identifies the [Queue] identified by the *QueuePathName* instance variable.

3.10.4.1.8 FormatName (Opnum 14)

The **FormatName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *QueueFormatName* instance variable, which contains the format name of the *referenced queue*.

```
[propget] HRESULT FormatName(  
    [out, retval] BSTR* pbstrFormatName  
);
```

pbstrFormatName: A pointer to BSTR that specifies the format name of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pbstrFormatName* output parameter to the value of the *QueueFormatName* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.9 FormatName (Opnum 15)

The **FormatName** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *QueueFormatName* instance variable, which contains the format name of the *referenced queue*.

```
[propput] HRESULT FormatName(  
    [in] BSTR bstrFormatName  
);
```

bstrFormatName: A BSTR that specifies the format name of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *QueueFormatName* instance variable to the *bstrFormatName* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.10 IsTransactional (Opnum 16)

The **IsTransactional** method is received by the server in an RPC_REQUEST packet. In response, the server returns a value that indicates whether the *referenced queue* is transactional or non-transactional.

```
[propget] HRESULT IsTransactional(  
    [out, retval] short* pisTransactional  
);
```

pisTransactional: A pointer to a short that corresponds to one of the [MQTRANSACTIONAL](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the IsTransactional instance variable is True
 - set the pisTransactional output parameter to MQ_TRANSACTIONAL (0x0001)
- otherwise
 - set the pisTransactional output parameter to MQ_TRANSACTIONAL_NONE (0x0000).

3.10.4.1.11 PrivLevel (Opnum 17)

The **PrivLevel** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *PrivacyLevel* instance variable, which specifies the privacy level of the *referenced queue*.

```
[propget] HRESULT PrivLevel(  
    [out, retval] long* plPrivLevel  
);
```

plPrivLevel: A pointer to a long that corresponds to one of the [MQPRIVLEVEL \(section 2.2.2.7\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pIPrivLevel* output parameter to the value of the *PrivacyLevel* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.12 PrivLevel (Opnum 18)

The **PrivLevel** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *PrivacyLevel* instance variable, which specifies the privacy level of the *referenced queue*.

```
[propput] HRESULT PrivLevel(
    [in] long lPrivLevel
);
```

IPrivLevel: A long that corresponds to one of the [MQPRIVLEVEL \(section 2.2.2.7\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *PrivacyLevel* instance variable to the value of the *IPrivLevel* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.13 Journal (Opnum 19)

The **Journal** method is received by the server in an RPC_REQUEST packet. In response, the server returns a value that specifies the journaling level of the *referenced queue*.

```
[propget] HRESULT Journal(
    [out, retval] long* plJournal
);
```

plJournal: A pointer to a long that corresponds to one of the [MQJOURNAL \(section 2.2.2.4\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *IsTargetJournalingEnabled* instance variable equals False:
 - Set the *plJournal* output parameter to the MQ_JOURNAL_NONE (0x00000000) value of the **MQJOURNAL** enumeration.
- Else:
 - Set the *plJournal* output parameter to the MQ_JOURNAL (0x00000001) value of the **MQJOURNAL** enumeration.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.14 Journal (Opnum 20)

The **Journal** method is received by the server in an RPC_REQUEST packet. In response, the server sets a value that specifies the journaling level of the *referenced queue*.

```
[propput] HRESULT Journal(  
    [in] long lJournal  
);
```

lJournal: A long that corresponds to one of the [MQJOURNAL \(section 2.2.2.4\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *lJournal* input parameter equals MQ_JOURNAL_NONE (0x00000000):
 - Set the *IsTargetJournalingEnabled* instance variable to FALSE (0x00000000).
- Else:
 - Set the *IsTargetJournalingEnabled* instance variable to TRUE (0x00000001).
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.15 Quota (Opnum 21)

The **Quota** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *Quota* instance variable, which specifies the maximum size (in kilobytes) of the *referenced queue*.

```
[propget] HRESULT Quota(  
    [out, retval] long* plQuota  
);
```

plQuota: A pointer to a long that specifies the maximum size (in kilobytes) of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *plQuota* output parameter to the value of the *Quota* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.16 Quota (Opnum 22)

The **Quota** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *Quota* instance variable that specifies the maximum size (in kilobytes) of the *referenced queue*.

```
[propput] HRESULT Quota(
    [in] long lQuota
);
```

lQuota: A long that specifies the maximum size (in kilobytes) of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *Quota* instance variable to the value of the *lQuota* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.17 BasePriority (Opnum 23)

The **BasePriority** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *BasePriority* instance variable, which specifies the queue-to-queue transfer priority of the *referenced queue*.

```
[propget] HRESULT BasePriority(
    [out, retval] long* plBasePriority
);
```

plBasePriority: A pointer to a long that specifies the queue-to-queue transfer priority of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *plBasePriority* output parameter to the value of the *BasePriority* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.18 BasePriority (Opnum 24)

The **BasePriority** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *BasePriority* instance variable, which specifies the queue-to-queue transfer priority of the *referenced queue*.

```
[propput] HRESULT BasePriority(
    [in] long lBasePriority
);
```

lBasePriority: A long that specifies the queue-to-queue transfer priority of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *BasePriority* instance variable to the value of the *IBasePriority* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.19 CreateTime (Opnum 25)

The **CreateTime** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *CreationTime* instance variable, which specifies the date and time when the *referenced queue* was created.

```
[propget] HRESULT CreateTime(  
    [out, retval] VARIANT* pvarCreateTime  
);
```

pvarCreateTime: A pointer to a VARIANT that contains a UTC (Coordinated Universal Time) date/time (VT_DATE) that specifies the date and time when the *referenced queue* was created.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pvarCreateTime* output parameter to the value of the *CreationTime* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.20 ModifyTime (Opnum 26)

The **ModifyTime** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *ModificationTime* instance variable, which specifies the latest date and time when one of properties of the *referenced queue* was updated.

```
[propget] HRESULT ModifyTime(  
    [out, retval] VARIANT* pvarModifyTime  
);
```

pvarModifyTime: A pointer to a VARIANT that contains a UTC date/time (VT_DATE) that specifies the latest date and time when one of properties of the *referenced queue* was updated.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pvarModifyTime* output parameter to the value of the *ModificationTime* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.21 Authenticate (Opnum 27)

The **Authenticate** method is received by the server in an RPC_REQUEST packet. In response, the server returns the authentication level for the *referenced queue*.

```
[propget] HRESULT Authenticate(  
    [out, retval] long* plAuthenticate  
);
```

plAuthenticate: A pointer to a long that corresponds to one of the [MQAUTHENTICATE \(section 2.2.2.6\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *AuthenticationRequired* instance variable equals True:
 - Set the *plAuthenticate* output parameter to MQ_AUTHENTICATE (0x00000001).
- Else:
 - Set *plAuthenticate* output parameter to MQ_AUTHENTICATE_NONE (0x00000000).
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.22 Authenticate (Opnum 28)

The **Authenticate** method is received by the server in an RPC_REQUEST packet. In response, the server sets the authentication level for the *referenced queue*.

```
[propput] HRESULT Authenticate(  
    [in] long lAuthenticate  
);
```

lAuthenticate: A long that corresponds to one of the [MQAUTHENTICATE \(section 2.2.2.6\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *lAuthenticate* input parameter equals MQ_AUTHENTICATE_NONE (0x00000000):
 - Set the *AuthenticationRequired* instance variable to FALSE (0x00000000).
- Else:
 - Set the *AuthenticationRequired* instance variable to TRUE (0x00000001).
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.23 JournalQuota (Opnum 29)

The **JournalQuota** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *JournalQuota* instance variable, which specifies the maximum size, in kilobytes, that is allowed for the [Journal Queue].[Messages Table] of the *referenced queue*.

```
[propget] HRESULT JournalQuota(  
    [out, retval] long* plJournalQuota  
);
```

plJournalQuota: A pointer to a long that specifies the maximum size, in kilobytes, allowed for the [Journal Queue].[Messages Table] of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *plJournalQuota* output parameter to the value of the *JournalQuota* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.24 JournalQuota (Opnum 30)

The **JournalQuota** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *JournalQuota* instance variable, which specifies the maximum size, in kilobytes, that is allowed for the [Journal Queue].[Messages Table] of the *referenced queue*.

```
[propput] HRESULT JournalQuota(  
    [in] long lJournalQuota  
);
```

lJournalQuota: A long that specifies the maximum size, in kilobytes, allowed for the [Journal Queue].[Messages Table] of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *JournalQuota* instance variable to the value of the *lJournalQuota* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.25 IsWorldReadable (Opnum 31)

The **IsWorldReadable** method is received by the server in an RPC_REQUEST packet. In response, the server returns an *IsWorldReadable* instance variable that indicates whether the *referenced queue* is accessible to everyone, or only to the owner and the system administrators.

```
[propget] HRESULT IsWorldReadable(  
    [out, retval] short* pisWorldReadable  
);
```


pisWorldReadable: A pointer to a short that indicates whether the *referenced queue* is accessible to everyone, or only to the owner and the system administrators.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pisWorldReadable* output parameter to the value of the *IsWorldReadable* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.26 Create (Opnum 32)

The **Create** method is received by the server in an RPC_REQUEST packet. In response, the server creates a new public or private [Application Queue].

```
HRESULT Create(  
    [in, optional] VARIANT* IsTransactional,  
    [in, optional] VARIANT* IsWorldReadable  
);
```

IsTransactional: A VARIANT pointer to a Boolean value (VT_BOOL) that specifies whether the queue is transactional. If the value is TRUE (0x00000001), the queue is transactional. If the value is FALSE (0x00000000), the queue is not transactional. If the value is unspecified, the server MUST assume that this value is FALSE.

IsWorldReadable: A VARIANT pointer to a Boolean value (VT_BOOL), that if set to TRUE (0x00000001) specifies that the queue is accessible to everyone. If the value is not specified, the server MUST use FALSE (0x00000000), which specifies that the queue will only be accessible to the owner and system administrators.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *QueuePathName* instance variable equals NULL:
 - Return an error HRESULT and take no further action.
- Identify the computer name and the queue name from the *QueuePathName* instance variable.
- If the computer name or queue name cannot be identified:
 - Return an error HRESULT and take no further action.
- Define *IsLocal* as a Boolean value that equals True if the [Computer Name] of the [Local Queue Manager] equals the identified computer name.
- Define *IsPublic* as a Boolean value that equals True if the *QueuePathName* instance variable describes a public queue. Otherwise, *IsPublic* equals False.
- If *IsLocal* equals False and *IsPublic* equals False:

- Return an error HRESULT and take no further action.
- Define *OwnerQueueManager* as a [Queue Manager] that equals:
 - [Local Queue Manager] if *IsLocal* equals True.
 - [Queue Manager] for which the [Computer Name] property equals the identified computer name.
- Check whether the *OwnerQueueManager* has an [Application Queue] for which the [Name] property equals the identified queue name.
- If [Application Queue] exists:
 - Return an error HRESULT and take no further action.
- Else:
 - Create an [Application Queue], termed *NewQueue*, and set its properties to the values of the respective instance variables of this instance of the [MSMQQueueInfo](#) object, as described by the mappings in the abstract data model section for this object, and set the following properties on *NewQueue*:
 - Set [Name] to the value of the identified queue name.
 - Set [IsTransactional] to the value of the *IsTransactional* input parameter.
 - Set [Owner] to the SID of the caller. [<50>](#)
 - Set [IsWorldReadable] to the value of the *IsWorldReadable* input parameter.
 - Add *NewQueue* to the [Queues Table] of the *OwnerQueueManager*.
 - Set the *QueueFormatName* instance variable to a format name that identifies the *NewQueue*.
- If *IsPublic* equals True:
 - Identify the [Directory Queue Manager] in the [Directory Queue Managers Table] of the [Directory] for which the [Computer Name] property of the [Directory Queue Manager] equals the identified computer name.
 - If the [Directory Queue Manager] cannot be identified:
 - Return an error HRESULT and take no further action.
 - Create a [Directory Queue] that exposes *NewQueue*.
 - Add the newly created [Directory Queue] to the [Directory Queues Table] of the identified [Directory Queue Manager].
- Refresh all the read-only instance variables to the values of the *NewQueue*'s properties, using the mapping described in the abstract data model section of the *MSMQQueueInfo* object.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.27 Delete (Opnum 33)

The **Delete** method is received by the server in an RPC_REQUEST packet. In response, the server deletes the referenced *queue*.

```
HRESULT Delete();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *QueueFormatName* instance variable is NULL:
 - Return an error HRESULT and take no further action.
- If the *QueueFormatName* instance variable identifies more than one queue or contains an HTTP or multicast format name:
 - Return an error HRESULT and take no further action.
- Look up the [Application Queue] identified by the *QueueFormatName* instance variable.
 - If [Application Queue] does not exist:
 - Return an error HRESULT and take no further action.
- Define *IsLocal* as a Boolean value that equals True if the identified [Application Queue] belongs to the [Queues Table] of the [Local Queue Manager]. Otherwise, *IsLocal* equals False.
- Define *IsPublic* as a Boolean value that equals True if the identified [Application Queue] is a [Public Queue]. Otherwise, *IsPublic* equals False.
- If *IsLocal* equals False and *IsPublic* equals False:
 - Return an error HRESULT and take no further action.
- Remove the identified [Application Queue] from the [Queues Table] of the respective [Queue Manager].
- If *IsPublic* equals True
 - Identify the [Directory Queue Manager] in the [Directory Queue Managers Table] of the [Directory] for which the [Computer Name] property of the [Directory Queue Manager] equals the [Computer Name] property of the [Queue Manager] that has the identified [Application Queue].
 - If the [Directory Queue Manager] cannot be identified:
 - Return error HRESULT and take no further action.
 - Identify the [Directory Queue] in the identified [Directory Queue Manager] for which the [Name] property of the [Directory Queue] equals the [Name] property of the identified [Application Queue].

- If [Directory Queue] exists
 - Remove the identified [Directory Queue] from the [Directory Queues Table] of the identified [Directory Queue Manager].
- Else
 - Return an error HRESULT and take no further action.
- Reset all the instance variables to the values described in the initialization section of the MSMQQueueInfo object.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.28 Open (Opnum 34)

The **Open** method is received by the server in an RPC_REQUEST packet. In response, the server opens the *referenced queue*.

```
HRESULT Open(
    [in] long Access,
    [in] long ShareMode,
    [out, retval] IMSMQQueue4** ppqueue
);
```

Access: A long that corresponds to one of the [MQACCESS \(section 2.2.2.3\)](#) enumeration values.

ShareMode: A long that corresponds to one of the [MQSHARE \(section 2.2.2.2\)](#) enumeration values.

ppqueue: A pointer to an [IMSMQQueue4](#) interface pointer that the server MUST set with an instance object of [MSMQQueue](#) that represents the open queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *QueueFormatName* instance variable is NULL:
 - Return an error HRESULT and take no further action.
- If the *Access* input parameter value is equal to MQ_SEND_ACCESS (0x00000002), and the *ShareMode* input parameter value is not equal to MQ_DENY_NONE (0x00000000):
 - Return an error HRESULT and take no further action.
- If the *QueueFormatName* instance variable contains an HTTP or multicast format name, or identifies more than one queue, and the *Access* input parameter is not equal to MQ_SEND_ACCESS (0x00000002):
 - Return an error HRESULT and take no further action.

- Define *IsLocal* as a Boolean value that equals True if the *QueueFormatName* instance variable identifies a [Queue] that belongs to the [Queues Table] of the [Local Queue Manager]. Otherwise, *IsLocal* is False.
- Define *IsPublic* as a Boolean value that equals True if the *QueueFormatName* instance variable identifies a [Public Queue]. Otherwise, *IsPublic* equals False.
- Define *SendingToOutgoing* and *ReceivingFromOutgoing*, both as Boolean values with initial values set to False.
- If *IsLocal* equals True:
 - If the Access input parameter is equal to (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS) (0x00000081) or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) (0x000000a0):
 - Set *ReceivingFromOutgoing* to True.
 - If (*IsLocal* equals False) or (the *QueueFormatName* instance variable either contains an HTTP or multicast format name, or identifies more than one queue):
 - If the Access input parameter is equal to MQ_SEND_ACCESS (0x00000002):
 - Set *SendingToOutgoing* to True.
- If *SendingToOutgoing* equals False and *ReceivingFromOutgoing* equals False:
 - Look up the [Queue] identified by the *QueueFormatName* instance variable, and term it as the *TargetQueue*.
 - If [Queue] does not exist:
 - Return an error HRESULT and take no further action.
- If *SendingToOutgoing* equals True or if *ReceivingFromOutgoing* equals True:
 - Look up the [Outgoing Queue] in the [Queues Table] of the [Local Queue Manager] for which the [Destination Format Name] property equals *QueueFormatName*.
 - If [Outgoing Queue] is found, term it as the *TargetQueue*.
 - If [Outgoing Queue] doesn't exist:
 - If *SendingToOutgoing* equals True:
 - Create an [Outgoing Queue], term it as the *TargetQueue*, and set the following property:
 - Set [Destination Format Name] to the value of the *QueueFormatName* instance variable.
 - Add the created [Outgoing Queue] to the [Queues Table] of the [Local Queue Manager].
 - Else:
 - Return an error HRESULT and take no further action.
- Define the following states of the *TargetQueue*:

- *Closed*: There is no [Open Queue] in the [Open Queues Table] of the [Local Queue Manager] with a [Queue Reference] that identifies the *TargetQueue*.
- *Open Shared*: There are one or more [Open Queue]s whose [IsExclusive] property equals False in the [Open Queues Table] of the [Local Queue Manager] with a [Queue Reference] that identifies the *TargetQueue*.
- *Opened Exclusively*: There is one [Open Queue] whose [IsExclusive] property equals True in the [Open Queues Table] of the [Local Queue Manager] with a [Queue Reference] that identifies the *TargetQueue*.
- Define *Result* according to the following table.

State of TargetQueue	ShareMode input parameter value	Result
Closed	Any	Success
Opened Exclusively	Any	Failure
Open Shared	MQ_DENY_RECEIVE_SHARE	Failure
Open Shared	MQ_DENY_NONE	Success

- If the *Result* is *Success*:
 - Create an [Open Queue] that represents the *TargetQueue*, and set the following properties:
 - Set [IsExclusive] to True if the *ShareMode* input parameter equals MQ_DENY_RECEIVE_SHARE; otherwise set it to False.
 - Set [Access] to the value of the *Access* input parameter.
 - Set [Queue Reference] to the *TargetQueue*.
 - Add the created [Open Queue] to the [Open Queues Table] of the [Local Queue Manager].
 - Instantiate an instance of MSMQQueue and initialize it with the created [Open Queue].
 - Set the *ppqueue* output parameter to the instantiated MSMQQueue instance.
 - Transition the object state to *Opened* state.
- Else:
 - Return an error HRESULT and take no further action.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.29 Refresh (Opnum 35)

The **Refresh** method is received by the server in an RPC_REQUEST packet. In response, the server refreshes the properties of the [MSMQQueueInfo](#) object with the values stored in the [Directory] (for public queues), or the [Local Queue Manager] (for private queues).

```
HRESULT Refresh();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *QueueFormatName* instance variable is NULL:
 - Return an error HRESULT and take no further action.
- If the *QueueFormatName* instance variable identifies more than one queue, or contains an HTTP or multicast format name:
 - Return an error HRESULT and take no further action.
- Define *IsLocal* as a Boolean value that equals True if the [Application Queue] identified by the *QueueFormatName* instance variable belongs to the [Queues Table] of the [Local Queue Manager]. Otherwise, *IsLocal* is False.
- Define *IsPublic* as a Boolean value that equals True if the *QueueFormatName* instance variable identifies a [Public Queue]. Otherwise, *IsPublic* equals False.
- If *IsLocal* equals False and *IsPublic* equals False:
 - Return an error HRESULT and take no further action.
- Look up the [Application Queue] identified by the *QueueFormatName* instance variable.
 - If [Application Queue] exists:
 - If *IsPublic* equals True:
 - Identify the [Directory Queue Manager] in the [Directory Queue Managers Table] of the [Directory] for which the [Computer Name] property of the [Directory Queue Manager] equals the [Computer Name] property of the [Queue Manager] of the identified [Application Queue].
 - If the [Directory Queue Manager] cannot be identified:
 - Return an error HRESULT and take no further action.
 - Identify the [Directory Queue] in the [Directory Queues Table] of the identified [Directory Queue Manager] for which the [Name] property of the [Directory Queue] equals the [Name] property of the identified [Application Queue].
 - If [Directory Queue] exists:
 - Retrieve the identified [Directory Queue]'s properties, and set the instance variables of the MSMQQueueInfo object to the values of the retrieved properties, according to the mappings described in the abstract data model section for this object, and, transitively, according to the definition of the [Directory Queue] in the system abstract data model in section [3.1.1](#).
 - Else:

- Retrieve the identified [Application Queue]'s properties, and set the instance variables of the MSMQQueueInfo object to the values of the retrieved properties, according to the mappings described in the abstract data model section for this object.
- Else:
 - Return an error HRESULT and take no further action.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.30 Update (Opnum 36)

The **Update** method is received by the server in an RPC_REQUEST packet. In response, the server updates the [Directory] or the [Local Queue Manager] with the current values of the [MSMQQueueInfo](#) object's properties.

```
HRESULT Update();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *QueueFormatName* instance variable is NULL:
 - Return an error HRESULT and take no further action.
- If the *QueueFormatName* instance variable identifies more than one queue, or contains an HTTP or multicast format name:
 - Return an error HRESULT and take no further action.
- Define *IsLocal* as a Boolean value that equals True if the [Application Queue] identified by *QueueFormatName* instance variable belongs to the [Queues Table] of the [Local Queue Manager]. Otherwise, *IsLocal* equals False.
- Define *IsPublic* as a Boolean value that equals True if the *QueueFormatName* instance variable identifies a [Public Queue]. Otherwise, *IsPublic* equals False.
- If *IsLocal* equals False and *IsPublic* equals False:
 - Return an error HRESULT and take no further action.
- Look up the [Application Queue] that is identified by the *QueueFormatName* instance variable.
 - If [Application Queue] exists:
 - Update the identified [Application Queue]'s properties with the instance variables of the MSMQQueueInfo object, according to the mappings described in the abstract data model section for this object.
 - If *IsPublic* equals True:
 - Identify the [Directory Queue Manager] in the [Directory Queue Managers Table] of the [Directory] for which the [Computer Name] property of the [Directory Queue Manager]

equals the [Computer Name] property of the [Queue Manager] of the identified [Application Queue].

- Identify the [Directory Queue] in the [Directory Queues Table] of the Identified [Directory Queue Manager] for which the [Name] property of the [Directory Queue] equals the [Name] property of the identified [Application Queue].
 - If [Directory Queue] exists:
 - Update the identified [Directory Queue]'s properties with the instance variables of the MSMQQueueInfo object. according to the mappings described in the abstract data model for this object, and, transitively, according to the definition of the [Directory Queue] in the system abstract data model in section [3.1.1](#).
 - Else:
 - Return an error HRESULT and take no further action.
- Else:
 - Return an error HRESULT and take no further action.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.31 PathNameDNS (Opnum 37)

The **PathNameDNS** method is received by the server in an RPC_REQUEST packet. In response, the server returns the DNS path name that identifies the *referenced queue*.

```
[propget] HRESULT PathNameDNS(  
    [out, retval] BSTR* pbstrPathNameDNS  
);
```

pbstrPathNameDNS: A pointer to a BSTR that specifies the DNS path name of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *QueueFormatName* instance variable is NULL:
 - Return an error HRESULT and take no further action.
- If the *QueueFormatName* instance variable identifies more than one queue, or contains an HTTP or multicast format name:
 - Return an error HRESULT and take no further action.
- Look up the [Queue] identified by the *QueueFormatName* instance variable.
 - If [Queue] exists:
 - If [Queue] is a [Public Queue]

- Construct the path name using the fully qualified **DNS name** for the [Computer Name] property of the identified [Public Queue]'s [Queue Manager] and the [Name] property of the identified [Public Queue], and term it as *DNSPathName*.
- Else:
 - Return an error HRESULT and take no further action.
- Else:
 - Return an error HRESULT and take no further action.
- Set the *pbstrPathNameDNS* output parameter to the value of *DNSPathName*.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.32 Properties (Opnum 38)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(
    [out, retval] IDispatch** ppcolProperties
);
```

ppcolProperties: A pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The server MUST take no action and return E_NOTIMPL (0x80004001).

3.10.4.1.33 Security (Opnum 39)

The **Security** method is not implemented.

```
[propget] HRESULT Security(
    [out, retval] VARIANT* pvarSecurity
);
```

pvarSecurity: A pointer to a VARIANT. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The get_security method MUST take no action and immediately return E_NOTIMPL (0x80004001).

3.10.4.1.34 Security (Opnum 40)

The **Security** method is not implemented.

```
[propput] HRESULT Security(
    [in] VARIANT varSecurity
);
```

varSecurity: The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The put_Security method MUST take no action and immediately return E_NOTIMPL (0x80004001).

3.10.4.1.35 IsTransactional2 (Opnum 41)

The **IsTransactional2** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *IsTransactional* instance variable, which indicates whether the referenced queue is transactional or non-transactional.

```
[propget] HRESULT IsTransactional2(  
    [out, retval] short* pisTransactional  
);
```

pisTransactional: A pointer to a VARIANT_BOOL that indicates whether the queue is transactional or non-transactional.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pisTransactional* output parameter to the value of the *IsTransactional* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.36 IsWorldReadable (Opnum 42)

The **IsWorldReadable** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *IsWorldReadable* instance variable, which indicates whether the *referenced queue* is accessible to everyone, or only to the owner and the system administrators.

```
[propget] HRESULT IsWorldReadable(  
    [out, retval] short* pisWorldReadable  
);
```

pisWorldReadable: A pointer to a VARIANT_BOOL that indicates whether the *referenced queue* is accessible to everyone, or only to the owner and the system administrators.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pisWorldReadable* output parameter to the value of the *IsWorldReadable* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.37 MulticastAddress (Opnum 43)

The **MulticastAddress** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *MulticastAddress* instance variable, which specifies the multicast address on which the *referenced queue* listens.

```
[propget] HRESULT MulticastAddress(
    [out, retval] BSTR* pbstrMulticastAddress
);
```

pbstrMulticastAddress: A pointer to a BSTR that specifies the multicast address on which the *referenced queue* listens.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *pbstrMulticastAddress* output parameter to the value of the *MulticastAddress* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.38 MulticastAddress (Opnum 44)

The **MulticastAddress** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *MulticastAddress* instance variable, which specifies the multicast address on which the *referenced queue* listens.

```
[propput] HRESULT MulticastAddress(
    [in] BSTR bstrMulticastAddress
);
```

bstrMulticastAddress: A BSTR that specifies the multicast address on which the *referenced queue* listens.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- Set the *MulticastAddress* instance variable to the value of the *bstrMulticastAddress* input parameter.
- Return S_OK (0x00000000) and take no further action.

3.10.4.1.39 ADsPath (Opnum 45)

The **ADsPath** method is received by the server in an RPC_REQUEST packet. In response, the server returns the directory path that identifies the *referenced queue*.

```
[propget] HRESULT ADsPath(
    [out, retval] BSTR* pbstrADsPath
);
```

pbstrADsPath: A pointer to a BSTR that contains the directory path of the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST abide by the following contract:

- If the *QueueFormatName* instance variable is NULL:
 - Return an error HRESULT and take no further action.
- If the *QueueFormatName* instance variable identifies more than one *queue*, or contains an HTTP or multicast format name:
 - Return an error HRESULT and take no further action.
- Look up the [Queue] that is identified by the *QueueFormatName* instance variable.
 - If [Queue] exists:
 - If [Queue] is a [Public Queue]:
 - Identify the [Directory Queue Manager] in the [Directory Queue Managers Table] of the [Directory] for which [Computer Name] property equals the [Computer Name] of the [Queue Manager] of the identified [Public Queue].
 - Identify the [Directory Queue] in the [Directory Queues Table] of the identified [Directory Queue Manager] for which the [Name] property equals the [Name] property of the identified [Public Queue].
 - Define *DirectoryPath* as the value of the [Path] property of the identified [Directory Queue].
 - Else:
 - Return an error HRESULT and take no further action.
 - Else:
 - Return an error HRESULT and take no further action.
- Set the *pbstrADsPath* output parameter to the value of the *DirectoryPath*.
- Return S_OK (0x00000000) and take no further action.

3.10.5 Timer Events

No timer events are required.

3.10.6 Other Local Events

No local events are required.

3.11 MSMQQueue Coclass

The MSMQQueue object represents an [Open Queue]. An [Open Queue] represents a server's granted permission to perform particular operations on a [Queue], termed here as the *referenced queue*.

A [Message] can be read from the [Messages Table] of the *referenced queue* in two ways:

- Peek, a non-destructive read operation that enables the client to read the [Message] without removing it from the [Messages Table] of the *referenced queue*.

- Receive, a destructive read operation that enables the client to read the [Message] and remove it from the [Messages Table] of the *referenced queue*.

The MSMQueue object implementation allows for receive operations to be performed within the scope of a transaction. The server MUST use the client input parameters and the execution context to get the transaction identifier for the transaction. For more details on how the transaction identifier is marshaled in DCOM, refer to [\[MS-DTCO\]](#) section 1.3.1.

Retrieving a [Message] from the [Messages Table] of the referenced [Queue] can be done using one of three approaches:

- Using a cursor, controlled through the *Cursor* instance variable.
- By searching for specific [Message]s in the [Messages Table] of the referenced [Queue], using a client-provided lookup identifier.
- Sequentially from the head of the referenced [Queue]'s [Messages Table].

3.11.1 Abstract Data Model

An implementation of the [MSMQQueue](#) coclass maintains the following abstract data elements:

- *OpenQueueReference*: A reference to the [Open Queue]. Formally, the term *referenced queue* is defined as the [Queue] that is identified by the [Queue Reference] property of the [Open Queue].
- *Cursor*: An internal cursor that is used while iterating over the [Message]s in the [Messages Table] of the *referenced queue*.
- *MSMQQueueInfoObject*: A pointer to the [MSMQQueueInfo](#) object instance that created and initialized this instance of the MSMQueue coclass.
- *IsClosed*: A Boolean value that, when equal to True, indicates that the object is in the closed state.
- *IsInitialized*: A Boolean value that, when equal to True, indicates that the object has been initialized by an MSMQueueInfo object.

The following state machine diagrams describe the different states and transitions for this object and its data.

3.11.1.1 Object State Machine

The [MSMQQueue](#) object has two states: Opened, which is the initial state of the object when it is obtained using the [MSMQQueueInfo::Open](#) method; and Closed.

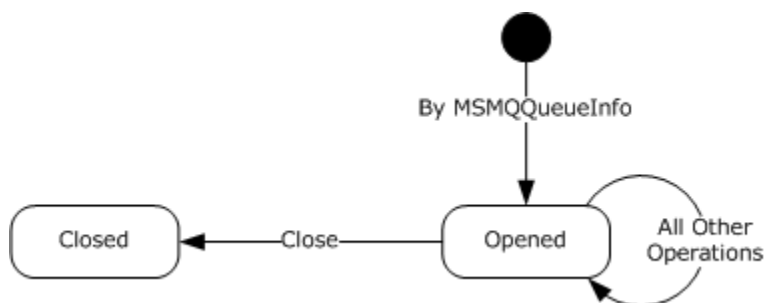


Figure 12: MSMQueue object states

Any call to any method in the object while it is in the *Closed* state MUST return MQ_ERROR_INVALID_HANDLE (0xC00E0007).

3.11.1.2 Cursor State Machine

The following diagram shows the state machine of the cursor that is represented by the *Cursor* instance variable:

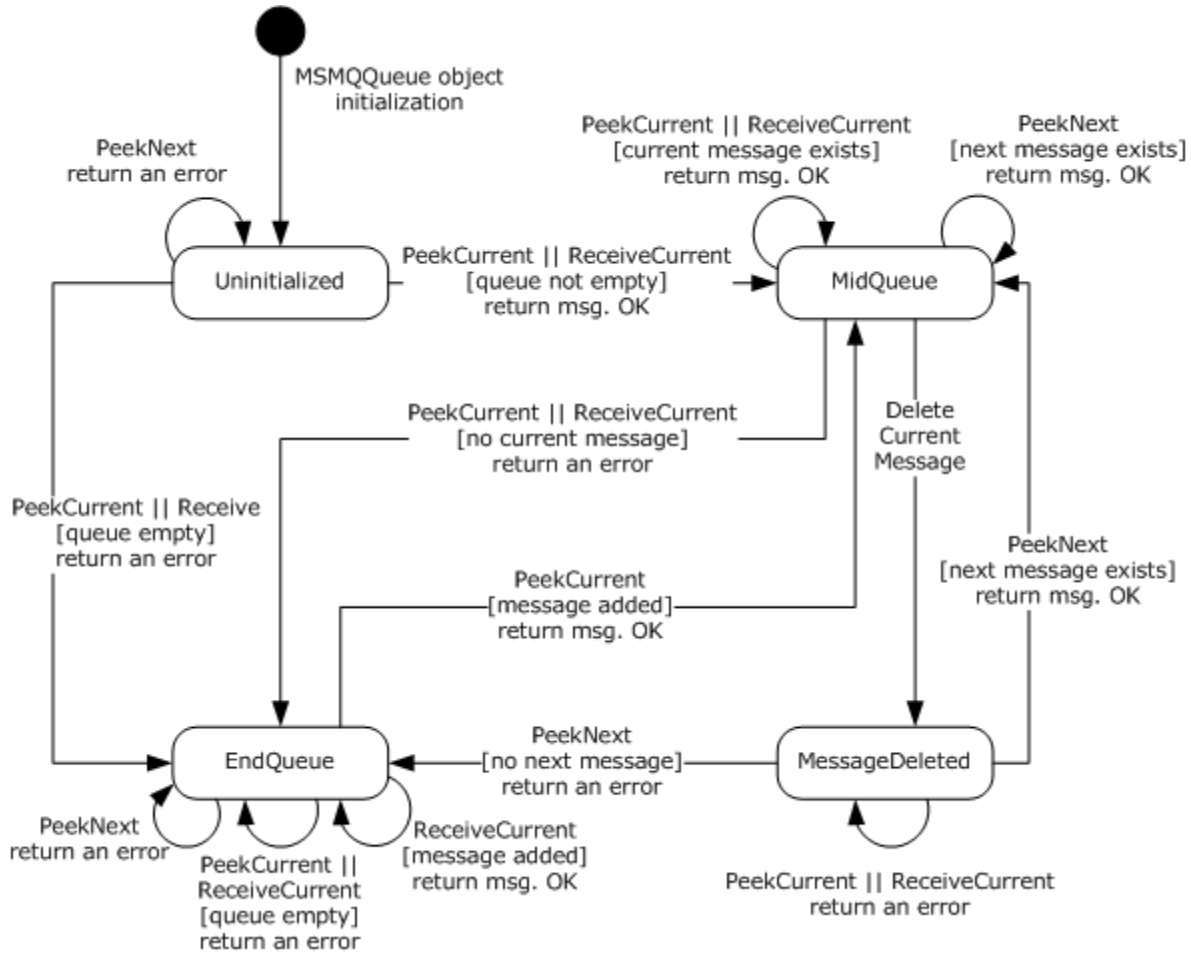


Figure 13: Cursor state diagram

3.11.2 Timers

The [MSMQQueue](#) object MUST maintain a timer with each call to receive or peek a [Message] from the associated queue, if the client supplied a timeout that is between 0 and INFINITE (0xFFFFFFFF).

3.11.3 Initialization

An [MSMQQueue](#) object MUST be instantiated and initialized by the server as a result of an invocation of the [MSMQQueueInfo::Open](#) method.

Initialization via **MSMQQueueInfo::Open** call results in the following:

- The *OpenQueueReference* instance variable is set to the [Open Queue] reference.
- The cursor represented by the *Cursor* instance variable is set to Uninitialized state.
- The *IsClosed* instance variable is set to False.
- The *IsInitialized* instance variable is set to True.
- The *MSMQQueueInfoObject* instance variable is set to the [MSMQQueueInfo](#) object instance that created this object instance.

If the client instantiated the MSMQQueue coclass directly, the server MUST initialize the object as follows:

- The *OpenQueueReference* instance variable is set to NULL.
- The *IsClosed* instance variable is set to True.
- The *IsInitialized* instance variable is set to False.

3.11.4 Message Processing Events and Sequencing Rules

This coclass includes four interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQQueue4	IMSMQQueue3	IMSMQQueue2	IMSMQQueue
7	get Access	X	X	X	X
8	get ShareMode	X	X	X	X
9	get QueueInfo	X	X	X	X
10	get Handle	X	X	X	X
11	get IsOpen	X	X	X	X
12	Close	X	X	X	X
13	Receive v1	X	X	X	X
14	Peek v1	X	X	X	X
15	EnableNotification	X	X	X	X
16	Reset	X	X	X	X
17	ReceiveCurrent v1	X	X	X	X
18	PeekNext v1	X	X	X	X
19	PeekCurrent v1	X	X	X	X
20	Receive	X	X	X	

Opnum	Method name (in the most recent interface revision)	IMSMQueue4	IMSMQueue3	IMSMQueue2	IMSMQueue
21	Peek	X	X	X	
22	ReceiveCurrent	X	X	X	
23	PeekNext	X	X	X	
24	PeekCurrent	X	X	X	
25	get Properties	X	X	X	
26	get Handle2	X	X		
27	ReceiveByLookupId	X	X		
28	ReceiveNextByLookupId	X	X		
29	ReceivePreviousByLookupId	X	X		
30	ReceiveFirstByLookupId	X	X		
31	ReceiveLastByLookupId	X	X		
32	PeekByLookupId	X	X		
33	PeekNextByLookupId	X	X		
34	PeekPreviousByLookupId	X	X		
35	PeekFirstByLookupId	X	X		
36	PeekLastByLookupId	X	X		
37	Purge	X	X		

Opnum	Method name (in the most recent interface revision)	IMSMQQueue4	IMSMQQueue3	IMSMQQueue2	IMSMQQueue
38	get IsOpen2	X	X		
39	ReceiveByLookupIdAllowPeak	X			

3.11.4.1 MSMQQueue4 Interface

The **IMSMQQueue4** interface provides methods that provide interaction with a queue on a given server. The version number for this interface is 4.0.

There are three previous versions of this interface: **IMSMQQueue**, **IMSMQQueue2**, and **IMSMQQueue3**. These previous versions are nearly identical, but have a few less methods. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server **MUST** implement a DCOM object class with the CLSID {d7d6e079-dccd-11d0-aa4b-0060970debae} (coclass **MSMQQueue** as specified in section 1.9), which implements the **IMSMQQueue4** interface using the UUID {eba96b20-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the **IUnknown** interface, as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the **IDispatch** interface, as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
Access (section 3.11.4.1.1) , get Access	Returns the access mode in which the queue was opened. Opnum: 7
ShareMode (section 3.11.4.1.2) , get SharedMode	Returns the share mode in which the queue was opened. Opnum: 8
QueueInfo (section 3.11.4.1.3) , get QueueInfo	Returns an MSMQQueueInfo object that contains the initial settings that were used to open the queue. <51> Opnum: 9
Handle (section 3.11.4.1.4) , get Handle	Returns the open queue's handle. Opnum: 10
IsOpen (section 3.11.4.1.5) , get IsOpen	Returns a value indicating whether the queue is open or not. Opnum: 11
Close (section 3.11.4.1.6)	Closes the open queue. Opnum: 12
Receive_v1 (section 3.11.4.1.7)	Retrieves the message at the head of the queue, and removes it from the queue. Opnum: 13

Method	Description
Peek v1 (section 3.11.4.1.8)	Retrieves the message at the head of the queue without removing it. Opnum: 14
EnableNotification (section 3.11.4.1.9)	Starts event notification for asynchronously reading messages in the queue. Opnum: 15
Reset (section 3.11.4.1.10)	Resets the cursor position to the head of the queue. Opnum: 16
ReceiveCurrent v1 (section 3.11.4.1.11)	Retrieves the message at the current cursor position, and removes it from the queue. Opnum: 17
PeekNext v1 (section 3.11.4.1.12)	Advances the cursor to the next message in the queue, and retrieves it without removing it. Opnum: 18
PeekCurrent v1 (section 3.11.4.1.13)	Retrieves the message at the current cursor position without removing it. Opnum: 19
Receive (section 3.11.4.1.14)	Retrieves the message at the head of the queue, and removes it from the queue. <52> <53> Opnum: 20
Peek (section 3.11.4.1.15)	Retrieves the message at the head of the queue without removing it. <54> <55> Opnum: 21
ReceiveCurrent (section 3.11.4.1.16)	Retrieves the message at the current cursor position, and removes it from the queue. <56> <57> Opnum: 22
PeekNext (section 3.11.4.1.17)	Advances the cursor to the next message in the queue, and retrieves it without removing it. <58> Opnum: 23
PeekCurrent (section 3.11.4.1.18)	Retrieves the message at the current cursor position without removing it. <59> <60> Opnum: 24
Properties (section 3.11.4.1.19) , get Properties	This method is not implemented. <61> Opnum: 25
Handle2 (section 3.11.4.1.20) , get Handle	Returns the handle of the open queue. <62> Opnum: 26
ReceiveByLookupId (section 3.11.4.1.21)	Retrieves the message with a lookup identifier that is equal to the specified lookup ID, and removes it from the queue. <63> <64> Opnum: 27

Method	Description
ReceiveNextByLookupId (section 3.11.4.1.22)	Retrieves the message that follows a message with a lookup identifier equal to the specified lookup ID, and removes it from the queue. <65> <66> Opnum: 28
ReceivePreviousByLookupId (section 3.11.4.1.23)	Retrieves the message that precedes a message with a lookup identifier equal to the specified lookup ID, and removes it from the queue. <67> <68> Opnum: 29
ReceiveFirstByLookupId (section 3.11.4.1.24)	Retrieves the message at the head of the queue, and removes it from the queue. <69> <70> Opnum: 30
ReceiveLastByLookupId (section 3.11.4.1.25)	Retrieves the message at the tail of the queue, and removes it from the queue. <71> <72> Opnum: 31
PeekByLookupId (section 3.11.4.1.26)	Retrieves the message with a lookup identifier that equals the specified lookup ID, without removing it. <73> <74> Opnum: 32
PeekNextByLookupId (section 3.11.4.1.27)	Retrieves the message that follows a message with a lookup identifier that equals the specified lookup ID, without removing it. <75> <76> Opnum: 33
PeekPreviousByLookupId (section 3.11.4.1.28)	Retrieves the message that precedes a message with a lookup identifier that equals the specified lookup ID, without removing it. <77> <78> Opnum: 34
PeekFirstByLookupId (section 3.11.4.1.29)	Retrieves the message at the head of the queue without removing it. <79> <80> Opnum: 35
PeekLastByLookupId (section 3.11.4.1.30)	Retrieves the message at the tail of the queue without removing it. <81> <82> Opnum: 36
Purge (section 3.11.4.1.31)	Deletes all the messages in the queue. <83> Opnum: 37
IsOpen2 (section 3.11.4.1.32) , get IsOpen2	Returns a value indicating whether the queue is open or not. <84> Opnum: 38
ReceiveByLookupIdAllowPeek (section 3.11.4.1.33)	Retrieves the message with a lookup identifier that equals the specified lookup ID, and removes it from the queue. <85> Opnum: 39

3.11.4.1.1 Access (Opnum 7)

The **Access** method is received by the server in an RPC_REQUEST packet. In response, the server returns a flag that indicates the access mode in which the queue was opened. The access mode specifies whether peek, receive, send, and/or administration operations can be performed.

```
[propget] HRESULT Access(  
    [out, retval] long* pAccess  
);
```

pAccess: A pointer to a long that corresponds to one of the [MQACCESS \(section 2.2.2.3\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- Set the *pAccess* output variable to the [Access Mode] property of the [Open Queue] that is referenced by the *OpenQueueReference* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.2 ShareMode (Opnum 8)

The **ShareMode** method is received by the server in an RPC_REQUEST packet. In response, the server returns a flag that indicates the share mode in which the queue was opened. The share mode specifies whether this instance of the open queue has an exclusive access to the queue or not.

```
[propget] HRESULT ShareMode(  
    [out, retval] long* pShareMode  
);
```

pShareMode: A pointer to a long that corresponds to one of the [MQSHARE \(section 2.2.2.2\)](#) enumeration values.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- If the [IsExclusive] property of the [Open Queue] referenced by *OpenQueueReference* instance variable equals True:
 - Set the *pShareMode* output variable to MQ_DENY_RECEIVE_SHARE (0x00000001).
- Else:

- Set the *plShareMode* output variable to MQ_DENY_NONE (0x00000000).
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.3 QueueInfo (Opnum 9)

The **QueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [MSMQQueueInfo](#) object that represents the *referenced queue*.

```
[propget] HRESULT QueueInfo(
    [out, retval] IMSMQQueueInfo** ppqinfo
);
```

ppqinfo: A pointer to an [IMSMQQueueInfo4](#) interface pointer that contains the **MSMQQueueInfo** object that represents the *referenced queue*.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- Set the *ppqinfo* output variable to *MSMQQueueInfoObject*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.4 Handle (Opnum 10)

The **Handle** method is received by the server in an RPC_REQUEST packet. In response, the server returns the handle of the open queue.

```
[propget] HRESULT Handle(
    [out, retval] long* plHandle
);
```

plHandle: A pointer to a long that represents the handle of the open queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- Set the *plHandle* output variable to the [Handle] property of the *OpenQueueReference* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.5 IsOpen (Opnum 11)

The **IsOpen** method is received by the server in an RPC_REQUEST packet. In response, the server returns a value that indicates whether the queue is open.

```
[propget] HRESULT IsOpen(  
    [out, retval] short* pisOpen  
);
```

pisOpen: A pointer to a short that MUST be set to TRUE (0x0001) if the queue is open, and FALSE (0x0000) if the queue is closed.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- If the *IsClosed* instance variable equals True:
 - Set the *pisOpen* output variable to FALSE (0x0000).
- Else:
 - Set the *pisOpen* output variable to TRUE (0x0001).
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.6 Close (Opnum 12)

The **Close** method is received by the server in an RPC_REQUEST packet. In response, the server closes an open instance of the queue.

```
HRESULT Close();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- Remove the [Open Queue] identified by the *OpenQueueReference* instance variable from the [Open Queues Table] of the *referenced queue*.
- Set the *OpenQueueReference* instance variable to NULL.
- Set the *IsClosed* instance variable to True.

- Return S_OK (0x00000000) and take no further action.

3.11.4.1.7 Receive_v1 (Opnum 13)

The **Receive_v1** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the head of the *referenced queue's* [Messages Table], and removes it.

```
HRESULT Receive_v1(
    [in, optional] VARIANT* Transaction,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [out, retval] IMMQMessage** ppmsg
);
```

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an [MSMQTransaction](#) object.
- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return a MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return a MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return a MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return a MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT containing a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

ppmsg: A pointer to an [IMSMQMessage](#) interface pointer that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the *Transaction* input parameter is not NULL:
 - If the *Transaction* input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the [ITransaction](#) instance that is identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the *Transaction* input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.1](#), using the enum numeric value represented by the *Transaction* input parameter.
 - Else:
 - Return an error and take no further action.
- Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
- If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction], and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] from here on.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].

- Define *suitable message* as the first [Message] from the head of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - If the *ReceiveTimeout* input parameter is 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - If the *ReceiveTimeout* input parameter is INFINITE (0xffffffff):
 - Block this call until a *suitable message* is available.
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details of initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
 - If the *ReceiveTimeout* input parameter is neither 0 nor INFINITE (0xffffffff):
 - Block the call until either of the following events occur:
 - The timeout specified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL.
 - Return MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
 - Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- If the *Transaction* input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:

- The [Message Reference] property with the *suitable message*.
- The [Operation Type] property to the [Operation Type].Receive enumeration value.
- Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] that is identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.8 Peek_v1 (Opnum 14)

The **Peek_v1** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the head of the *referenced queue*'s [Messages Table] without removing it.

```
HRESULT Peek_v1(
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [out, retval] MSMQMessage** ppmsg
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return a MSMQMessage object that has the Body property set.
VARIANT_FALSE	The server MUST return a MSMQMessage object that does not have the Body

Value	Meaning
0x0000	property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

ppmsg: A pointer to a pointer to an [IMSMQMessage](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- Define *suitable message* as the first [Message] from the head of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - If the *ReceiveTimeout* input parameter is 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - If the *ReceiveTimeout* input parameter is INFINITE (0xFFFFFFFF):
 - Block this call until a *suitable message* is available.
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and

WantDestinationQueue input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).

- Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- If the *ReceiveTimeout* input parameter is neither 0 nor INFINITE (0xFFFFFFFF):
 - Block the call until either of the following events occur:
 - The timeout specified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL.
 - Return MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
 - Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.9 EnableNotification (Opnum 15)

The **EnableNotification** method is received by the server in an RPC_REQUEST packet. In response, the server starts event notification for asynchronously receiving or peeking messages.

```
HRESULT EnableNotification(  
    [in] IMSMQEvent3* Event,  
    [in, optional] VARIANT* Cursor,  
    [in, optional] VARIANT* ReceiveTimeout  
);
```

Event: A pointer to an [IMSMQEvent3](#) interface. If *Event* is NULL, the server MUST return an error E_INVALIDARG (0x80070057).

Cursor: A VARIANT pointer to a signed integer that corresponds to one of the [MQMSGCURSOR \(section 2.2.2.8\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQMSG_FIRST (0x00000000).

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *Cursor* input parameter is not equal to MQMSG_FIRST, MQMSG_CURRENT, or MQMSG_NEXT:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- If the *Event* input parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the *Cursor* input parameter equals MQMSG_NEXT:
 - Advance the cursor represented by the *Cursor* instance variable.
- Return S_OK (0x00000000) and take no further action.

Asynchronously:

- If the *Cursor* input parameter equals MQMSG_NEXT or MQMSG_CURRENT:
 - Define *suitable message* as a [Message], identified by the cursor represented by the *Cursor* instance variable, in the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
 - Starting from the [Message] identified by the cursor represented by the *Cursor* instance variable, continually advance the cursor, seeking a *suitable message*. If the cursor reaches *EndQueue* state, wait for more messages to arrive. Do this until one of the following conditions occurs:
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter equals 0:
 - Fire the [ArrivedError \(section 3.16.4.3.2\)](#) event on the [MSMQEvent](#) object that is identified by the *Event* input parameter. Specify the *ErrorCode* parameter as MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008), and take no further action.

- No *suitable message* can be identified, and the *ReceiveTimeout* input parameter is greater than 0 and doesn't equal INFINITE, and the timeout identified by the *ReceiveTimeout* input parameter expires:
 - Fire the **ArrivedError** event on the MSMQEvent object identified by the *Event* input parameter. Specify the *ErrorCode* parameter as MQ_ERROR_IO_TIMEOUT (0xc00e001b), and take no further action.
- A *suitable message* becomes available:
 - Fire the **Arrived** (section 3.16.4.3.1) event on the MSMQEvent object identified by the *Event* input parameter, and take no further action.
- Else:
 - Define *suitable message* as the [Message] at the head of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
 - Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - If the *ReceiveTimeout* input parameter is 0:
 - Fire the **ArrivedError** event on the MSMQEvent object identified by the *Event* input parameter. Specify the *ErrorCode* parameter as MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008), and take no further action.
 - If the *ReceiveTimeout* input parameter is INFINITE (0xffffffff):
 - Wait until a *suitable message* is available.
 - Fire the **Arrived** event on the MSMQEvent object identified by the *Event* input parameter, and take no further action.
 - If the *ReceiveTimeout* input parameter is neither 0 nor INFINITE (0xffffffff):
 - Wait until either of the following events occur:
 - The timeout specified by the *ReceiveTimeout* input parameters expires:
 - Fire the **ArrivedError** event on the MSMQEvent object that is identified by the *Event* input parameter. Specify the *ErrorCode* parameter as MQ_ERROR_IO_TIMEOUT (0xc00e001b), and take no further action.
 - A *suitable message* becomes available:
 - Fire the **Arrived** event on the MSMQEvent object identified by the *Event* input parameter, and take no further action.
 - Else, if there are one or more *suitable messages* available:
 - Fire the **Arrived** event on the MSMQEvent object identified by the *Event* input parameter, and take no further action.

3.11.4.1.10 Reset (Opnum 16)

The **Reset** method is received by the server in an RPC_REQUEST packet. In response, the server resets the cursor that is represented by the *Cursor* instance variable to *Uninitialized* state.

```
HRESULT Reset();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- Set the cursor that is represented by the *Cursor* instance variable to *Uninitialized* state.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.11 ReceiveCurrent_v1 (Opnum 17)

The **ReceiveCurrent_v1** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the current cursor position in the *referenced queue's* [Messages Table], removes it, and advances the cursor.

```
HRESULT ReceiveCurrent_v1(  
    [in, optional] VARIANT* Transaction,  
    [in, optional] VARIANT* WantDestinationQueue,  
    [in, optional] VARIANT* WantBody,  
    [in, optional] VARIANT* ReceiveTimeout,  
    [out, retval] IMSMQMessage** ppmsg  
);
```

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an [MSMQTransaction](#) object.
- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT containing a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

ppmsg: A pointer to a pointer to an [IMSMQMessage](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the *Transaction* input parameter is not NULL:
 - If the *Transaction* input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object:

- Define *transaction identifier* as the unique transaction identifier obtained from the [ITransaction](#) instance identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
- Else, if the *Transaction* input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the *Transaction* input parameter.
- Else:
 - Return an error and take no further action.
- Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
- If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction] and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] henceforth.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
- If the message represented by the *Cursor* instance variable is in the *MessageDeleted* state:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_ALREADY_RECEIVED (0xc00e001d), and take no further action.
- Define *suitable message* as a [Message], identified by the cursor represented by the *Cursor* instance variable, in the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False.
- Starting from the [Message] identified by the cursor represented by the *Cursor* instance variable, continually advance the cursor, seeking a *suitable message*. If the cursor reaches *EndQueue* state, wait for more messages to arrive. Do this until one of the following conditions occurs:
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter equals 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter is greater than 0 and is not equal to INFINITE, and the timeout identified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
- A *suitable message* becomes available:

- Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
- Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- Advance the cursor represented by the *Cursor* instance variable.
- If the *Transaction* input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an enlisted MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.12 PeekNext_v1 (Opnum 18)

The **PeekNext_v1** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] that follows the [Message] that is identified by the cursor represented by the *Cursor* instance variable in the *referenced queue*'s [Messages Table], without removing it.

```
HRESULT PeekNext_v1(
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [out, retval] IMSMQMessage** ppmsg
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the state of the cursor that is identified by the *Cursor* instance variable equals *Uninitialized* or *EndQueue*:

- Return error MQ_ERROR_ILLEGAL_CURSOR_ACTION (0xC00E001C) and take no further action.
- Define *suitable message* as a [Message], identified by the cursor represented by the *Cursor* instance variable, in the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False, or [AllowPeekWhenLocked] equals True.
- Advance the cursor that is represented by the *Cursor* instance variable.
- Starting from the [Message] identified by the cursor represented by the *Cursor* instance variable, continually advance the cursor, seeking a *suitable message*. If the cursor reaches *EndQueue* state, wait for more messages to arrive. Do this until one of the following conditions occurs:
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter equals 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter is greater than 0 and is not equal to INFINITE, and the timeout identified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
 - Return S_OK (0x00000000) and take no further action.

3.11.4.1.13 PeekCurrent_v1 (Opnum 19)

The **PeekCurrent_v1** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] that is identified by the *Cursor* instance variable in the *referenced queue's* [Messages Table], without removing it.

```
HRESULT PeekCurrent_v1(
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [out, retval] IMSMQMessage** ppmsg
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines

- If *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the cursor represented by the *Cursor* instance variable is in the *MessageDeleted* state:
 - Set the *ppmsg* output parameter to NULL.

- Return an error MQ_ERROR_MESSAGE_ALREADY_RECEIVED (0xc00e001d), and take no further action.
- Define *suitable message* as a [Message], identified by the cursor that is represented by the *Cursor* instance variable, in the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
- Starting from the [Message] identified by the cursor represented by the *Cursor* instance variable, continually advance the cursor, seeking a *suitable message*. If the cursor reaches *EndQueue* state, wait for more messages to arrive. Do this until one of the following conditions occurs:
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter equals 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008), and take no further action.
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter is greater than 0 and is not equal to INFINITE, and the timeout identified by the *ReceiveTimeout* input parameter expires:
 - Set *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_IO_TIMEOUT (0xc00e001b), and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody* and *WantDestinationQueue* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
 - Return S_OK (0x00000000) and take no further action.

3.11.4.1.14 Receive (Opnum 20)

The **Receive** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the head of the *referenced queue*'s [Messages Table] and removes it.

```
HRESULT Receive(
    [in, optional] VARIANT* Transaction,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMMQMessage4** ppmsg
);
```

Transaction: A pointer to a VARIANT that MUST contain either:

A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an [MSMQTransaction](#) object.

A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the *Transaction* input parameter is not NULL:
 - If the *Transaction* input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an enlisted MSMQTransaction object:

Define *transaction identifier* as the unique transaction identifier obtained from the [ITransaction](#) instance identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the *Transaction* input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the *Transaction* input parameter.
 - Else:
 - Return an error and take no further action.
 - Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
 - If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction], and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] from here on.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
- Define *suitable message* as the first [Message] from the head of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:

- If the *ReceiveTimeout* input parameter is 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
- If the *ReceiveTimeout* input parameter is INFINITE (0xffffffff):
 - Block this call until a *suitable message* is available.
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- If the *ReceiveTimeout* input parameter is neither 0 nor INFINITE (0xffffffff):
 - Block the call until either of the following events occur:
 - The timeout specified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL.
 - Return MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
 - Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- If the *Transaction* input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation], and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transactional Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.

- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] that is identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.15 Peek (Opnum 21)

The **Peek** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the head of the *referenced queue*'s [Messages Table], without removing it.

```
HRESULT Peek(
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** pmsg
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- Define *suitable message* as the first [Message] from the head of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
- Attempt to retrieve the *suitable message*:
 - If no *suitable message* can be located:
 - If the *ReceiveTimeout* input parameter is 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - If the *ReceiveTimeout* input parameter is INFINITE (0xFFFFFFFF):

- Block this call until a *suitable message* is available.
- Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
- Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- If the *ReceiveTimeout* input parameter is neither 0 nor INFINITE (0xFFFFFFFF):
 - Block the call until either of the following events occur:
 - The timeout specified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL
 - Return MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.16 ReceiveCurrent (Opnum 22)

The **ReceiveCurrent** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] that is identified by the *Cursor* instance variable in the *referenced queue's* [Messages Table], and removes it.

```
HRESULT ReceiveCurrent(
    [in, optional] VARIANT* Transaction,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

Transaction: A pointer to a VARIANT that MUST contain either:

A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an [MSMQTransaction](#) object.

A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the *Transaction* input parameter is not NULL:
 - If the *Transaction* input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the ITransaction instance that is identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the *Transaction* input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the *Transaction* input parameter.
 - Else
 - Return an error and take no further action.
 - Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
 - If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction], and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] henceforth.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
- If the message represented by the *Cursor* instance variable is in the *MessageDeleted* state:
 - Set *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_ALREADY_RECEIVED (0xc00e001d) and take no further action.

- Define *suitable message* as a [Message], identified by the cursor represented by the *Cursor* instance variable, in the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False.
- Starting from the [Message] identified by the cursor represented by the *Cursor* instance variable, continually advance the cursor seeking a *suitable message*. If the cursor reaches *EndQueue* state, wait for more messages to arrive. Do this until one of the following conditions occurs:
 - No *suitable message* can be identified and ReceiveTimeout input parameter equals 0:
 - Set ppmsg output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - No *suitable message* can be identified and the ReceiveTimeout input parameter is greater than 0 and is not equal to INFINITE and the timeout identified by the ReceiveTimeout input parameter expires:
 - Set ppmsg output parameter to NULL.
 - Return an error MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message* and instantiate an MSMQMessage instance and initialize it with the *suitable message* observing the requirements set forth by the WantBody, WantDestinationQueue and WantConnectorType input parameters. For details of initializing the MSMQMessage object refer to section [3.17.3](#).
 - Set ppmsg output parameter to the newly instantiated MSMQMessage instance.
 - Advance the cursor represented by the *Cursor* instance variable.
 - If the Transaction input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
 - Else
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the referenced queue.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.17 PeekNext (Opnum 23)

The **PeekNext** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] that follows the [Message] that is identified by the *Cursor* instance variable in the *referenced queue's* [Messages Table], without removing it.

```
HRESULT PeekNext(  
    [in, optional] VARIANT* WantDestinationQueue,  
    [in, optional] VARIANT* WantBody,  
    [in, optional] VARIANT* ReceiveTimeout,  
    [in, optional] VARIANT* WantConnectorType,  
    [out, retval] MSMQMessage4** ppsmsg  
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE	The server MUST return an MSMQMessage object that has the

Value	Meaning
0xFFFF	ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the state of the cursor identified by the *Cursor* instance variable equals *Uninitialized* or *EndQueue*:
 - Return error MQ_ERROR_ILLEGAL_CURSOR_ACTION (0xC00E001C), and take no further action.
- Define *suitable message* as a [Message], identified by the cursor represented by the *Cursor* instance variable, in the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
- Advance the cursor that is represented by the *Cursor* instance variable.
- Starting from the [Message] identified by the cursor represented by the *Cursor* instance variable, continually advance the cursor seeking a *suitable message*. If the cursor reaches *EndQueue* state, wait for more messages to arrive. Do this until one of the following conditions occurs:
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter equals 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008), and take no further action.

- No *suitable message* can be identified, and the *ReceiveTimeout* input parameter is greater than 0 and is not equal to INFINITE, and the timeout identified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
- A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.18 PeekCurrent (Opnum 24)

The **PeekCurrent** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] that is identified by the *Cursor* instance variable in the *referenced queue's* [Messages Table], without removing it.

```
HRESULT PeekCurrent(
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* ReceiveTimeout,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.

Value	Meaning
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

ReceiveTimeout: A pointer to a VARIANT that contains a long value (VT_I4) that specifies the time, in milliseconds, that the server MUST not exceed while waiting for a new message to arrive.

If this parameter is not specified by the client, the server MUST use the default value INFINITE (0xFFFFFFFF).

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the *ppmsg* output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the message that is represented by the *Cursor* instance variable is in the *MessageDeleted* state:
 - Set the *ppmsg* output parameter to NULL.

- Return an error MQ_ERROR_MESSAGE_ALREADY_RECEIVED (0xc00e001d), and take no further action.
- Define *suitable message* as a [Message], identified by the cursor represented by the *Cursor* instance variable, in the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
- Starting from the [Message] identified by the cursor that is represented by the *Cursor* instance variable, continually advance the cursor seeking a *suitable message*. If the cursor reaches *EndQueue* state, wait for more messages to arrive. Do this until one of the following conditions occurs:
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter equals 0:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008), and take no further action.
 - No *suitable message* can be identified, and the *ReceiveTimeout* input parameter is greater than 0 and is not equal to INFINITE, and the timeout identified by the *ReceiveTimeout* input parameter expires:
 - Set the *ppmsg* output parameter to NULL.
 - Return an error MQ_ERROR_IO_TIMEOUT (0xc00e001b) and take no further action.
 - A *suitable message* becomes available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue*, and *WantConnectorType* input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated MSMQMessage instance.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.19 Properties (Opnum 25)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(
    [out, retval] IDispatch** ppcolProperties
);
```

ppcolProperties: A pointer to an IDispatch interface pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The server MUST take no action and immediately return E_NOTIMPL (0x80004001).

3.11.4.1.20 Handle2 (Opnum 26)

The **Handle2** method is received by the server in an RPC_REQUEST packet. In response, the server returns the handle of the open queue.

```
[propget] HRESULT Handle2(  
    [out, retval] VARIANT* pvarHandle  
);
```

pvarHandle: A pointer to a VARIANT of type VT_I8 (0x00000014) that represents the open queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST:

- If the *IsInitialized* instance variable is False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- Set the *pvarHandle* output variable to the [Handle] property of the [Open Queue] identified by the *OpenQueueReference* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.21 ReceiveByLookupId (Opnum 27)

The **ReceiveByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] from the *referenced queue's* [Messages Table] for which the [Lookup Identifier] property equals the lookup identifier, and removes it.

```
HRESULT ReceiveByLookupId(  
    [in] VARIANT LookupId,  
    [in, optional] VARIANT* Transaction,  
    [in, optional] VARIANT* WantDestinationQueue,  
    [in, optional] VARIANT* WantBody,  
    [in, optional] VARIANT* WantConnectorType,  
    [out, retval] IMSMQMessage4** ppmsg  
);
```

LookupId: A VARIANT (VT_UI8) that contains a value that represents a position in the [Messages Table] of the *referenced queue*, as described by the [Lookup Identifier] property of [Message].

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an enlisted [MSMQTransaction](#) object.
- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the LookupId input parameter equals 0:
 - Return an error HRESULT and take no further action.
- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:

- Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the Transaction input parameter is not NULL:
 - If the Transaction input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the ITransaction instance identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the Transaction input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the Transaction input parameter.
 - Else:
 - Return an error and take no further action.
 - Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
 - If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction], and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] from here on.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
 - Define *suitable message* as the [Message] at the position in the [Messages Table] of the *referenced queue* that is identified by the LookupId input parameter (as described by the [Lookup Identifier] of [Message]), where the [IsLocked] property of the [Message] equals False.
 - Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - Else:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the

WantConnectorType, WantBody and WantDestinationQueue input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).

- Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- If the Transaction input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.22 ReceiveNextByLookupId (Opnum 28)

The **ReceiveNextByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] following a [Message] from the *referenced queue*'s [Messages Table] for which the [Lookup Identifier] property equals the lookup identifier, and removes it.

```
HRESULT ReceiveNextByLookupId(  
    [in] VARIANT LookupId,  
    [in, optional] VARIANT* Transaction,  
    [in, optional] VARIANT* WantDestinationQueue,  
    [in, optional] VARIANT* WantBody,  
    [in, optional] VARIANT* WantConnectorType,  
    [out, retval] IMSMQMessage4** ppmsg  
);
```

LookupId: A VARIANT (VT_UI8) that contains a value that represents a position in the [Messages Table] of the *referenced queue*, as described by the [Lookup Identifier] property of [Message].

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an enlisted [MSMQTransaction](#) object.

- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If LookupId input parameter equals 0:
 - Return an error HRESULT and take no further action.

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the Transaction input parameter is not NULL:
 - If the Transaction input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the ITransaction instance that is identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the Transaction input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the Transaction input parameter.
 - Else:
 - Return an error and take no further action.
 - Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
 - If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction], and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] from here on.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
 - Define *suitable message* as the [Message] following the position in the [Messages Table] of the *referenced queue* that is identified by the LookupId input parameter (as described by the [Lookup Identifier] of [Message]), where the [IsLocked] property of the [Message] equals False.
 - Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.

- Else:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantConnectorType, WantBody and WantDestinationQueue input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- If the Transaction input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] that is identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.23 ReceivePreviousByLookupId (Opnum 29)

The **ReceivePreviousByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] preceding a [Message] from the *referenced queue*'s [Messages Table] for which the [Lookup Identifier] property equals the lookup identifier, and removes it.

```
HRESULT ReceivePreviousByLookupId(
    [in] VARIANT LookupId,
    [in, optional] VARIANT* Transaction,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

LookupId: A VARIANT (VT_UI8) that contains a value that represents a position in the [Messages Table] of the *referenced queue*, as described by the [Lookup Identifier] property of [Message].

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an enlisted [MSMQTransaction](#) object.
- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the LookupId input parameter equals 0:

- Return an error HRESULT and take no further action.
- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the Transaction input parameter is not NULL:
 - If the Transaction input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the ITransaction instance identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the Transaction input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the Transaction input parameter.
 - Else:
 - Return an error and take no further action.
 - Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
 - If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction], and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] from here on.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
 - Define *suitable message* as the [Message] preceding the position in the [Messages Table] of the *referenced queue* that is identified by the LookupId input parameter (as described by the [Lookup Identifier] of [Message]), where the [IsLocked] property of the [Message] equals False.
 - Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.

- Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
- Else:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantConnectorType, WantBody and WantDestinationQueue input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- If the Transaction input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] that is identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.24 ReceiveFirstByLookupId (Opnum 30)

The **ReceiveFirstByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the head of the *referenced queue*'s [Messages Table], and removes it.

```
HRESULT ReceiveFirstByLookupId(
    [in, optional] VARIANT* Transaction,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an [MSMQTransaction](#) object.
- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an [IMSMQMessage4](#) interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.

- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the Transaction input parameter is not NULL:
 - If the Transaction input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an enlisted MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the ITransaction instance identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the Transaction input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the Transaction input parameter.
 - Else:
 - Return an error and take no further action.
 - Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
 - If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction], and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] from here on.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
- Define *suitable message* as the first [Message] from the head of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xC00E0008) and take no further action.
 - Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantBody, WantDestinationQueue and WantConnectorType input parameters. For details on initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.

- If the Transaction input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] that is identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.25 ReceiveLastByLookupId (Opnum 31)

The **ReceiveLastByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the tail of the *referenced queue*'s [Messages Table], and removes it.

```
HRESULT ReceiveLastByLookupId(
    [in, optional] VARIANT* Transaction,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppsmsg
);
```

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object.
- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an IMSMQMessage4 interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:

- Return E_INVALIDARG (0x80070057) and take no further action.
- If the Transaction input parameter is not NULL:
 - If the Transaction input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an enlisted MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the ITransaction instance identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the Transaction input parameter is a VT_I4:
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the Transaction input parameter.
 - Else:
 - Return an error and take no further action.
- Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager] where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
- If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction] and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] henceforth.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager]
- Define *suitable message* as the [Message] at the tail of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantBody, WantDestinationQueue and WantConnectorType input parameters. For details of initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- If the Transaction input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:

- The [Message Reference] property with the *suitable message*.
- The [Operation Type] property to the [Operation Type].Receive enumeration value.
- Add the newly created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:
 - If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.26 PeekByLookupId (Opnum 32)

The **PeekByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] from the *referenced queue*'s [Messages Table] for which the [Lookup Identifier] property equals the lookup ID, without removing it.

```
HRESULT PeekByLookupId(
    [in] VARIANT LookupId,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] MSMQMessage4** pmsg
);
```

LookupId: A VARIANT (VT_UI8) that contains a value that represents a position in the [Messages Table] of the *referenced queue*, as described by the [Lookup Identifier] property of [Message].

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an IMSMQMessage4 interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If LookupId input parameter equals 0:
 - Return error HRESULT and take no further action.
- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- Define *suitable message* as the [Message] at the position in the [Messages Table] of the *referenced queue* identified by the LookupId input parameter (as described by the [Lookup Identifier] of [Message]), where the [IsLocked] property of the [Message] equals False or [AllowPeekWhenLocked] equals True.
- Attempt to retrieve the *suitable message*.

- If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
- Else:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantConnectorType, WantBody and WantDestinationQueue input parameters. For details of initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.27 PeekNextByLookupId (Opnum 33)

The **PeekNextByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] following a [Message] from the *referenced queue's* [Messages Table] for which the [Lookup Identifier] property equals the lookup identifier, without removing it.

```
HRESULT PeekNextByLookupId(
    [in] VARIANT LookupId,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

LookupId: A VARIANT (VT_UI8) that contains a value that represents a position in the [Messages Table] of the *referenced queue*, as described by the [Lookup Identifier] property of [Message].

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an IMSMQMessage4 interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the LookupId input parameter equals 0:
 - Return error HRESULT and take no further action.
- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- Define *suitable message* as the [Message] following the position in the [Messages Table] of the *referenced queue* identified by the LookupId input parameter (as described by the [Lookup Identifier] of [Message]), where the [IsLocked] property of the [Message] equals False or [AllowPeekWhenLocked] equals True.
- Attempt to retrieve the *suitable message*.

- If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
- Else:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantConnectorType, WantBody and WantDestinationQueue input parameters. For details of initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.28 PeekPreviousByLookupId (Opnum 34)

The **PeekPreviousByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] preceding a [Message] from the *referenced queue's* [Messages Table] for which the [Lookup Identifier] property equals the lookup identifier, without removing it.

```
HRESULT PeekPreviousByLookupId(
    [in] VARIANT LookupId,
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

LookupId: A VARIANT (VT_UI8) that contains a value that represents a position in the [Messages Table] of the *referenced queue* as described by the [Lookup Identifier] property of [Message].

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an IMSMQMessage4 interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the LookupId input parameter equals 0:
 - Return error HRESULT and take no further action.
- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- Define *suitable message* as the [Message] preceding the position in the [Messages Table] of the *referenced queue* identified by the LookupId input parameter (as described by the [Lookup Identifier] of [Message]), where the [IsLocked] property of the [Message] equals False or [AllowPeekWhenLocked] equals True.
- Attempt to retrieve the *suitable message*.

- If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
- Else:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantConnectorType, WantBody and WantDestinationQueue input parameters. For details of initializing the MSMQMessage object. refer to section [3.17.3](#).
- Set the ppmsg output parameter to the newly instantiated MSMQMessage instance. Return S_OK (0x00000000) and take no further action.

3.11.4.1.29 PeekFirstByLookupId (Opnum 35)

The **PeekFirstByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the head of the *referenced queue's* [Messages Table], without removing it.

```
HRESULT PeekFirstByLookupId(
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an IMSMQMessage4 interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to MQ_PEEK_ACCESS or (MQ_PEEK_ACCESS | MQ_ADMIN_ACCESS) or MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- Define *suitable message* as the first [Message] from the head of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals False or [AllowPeekWhenLocked] equals True.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantBody, WantDestinationQueue and WantConnectorType input parameters. For details of initializing the MSMQMessage object, refer to section [3.17.3](#).

- Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.30 PeekLastByLookupId (Opnum 36)

The **PeekLastByLookupId** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] at the tail of the *referenced queue's* [Messages Table], without removing it.

```
HRESULT PeekLastByLookupId(
    [in, optional] VARIANT* WantDestinationQueue,
    [in, optional] VARIANT* WantBody,
    [in, optional] VARIANT* WantConnectorType,
    [out, retval] IMSMQMessage4** ppmsg
);
```

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE 0x0000	The server MUST return an MSMQMessage object that does not have the Body property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an `IMSMQMessage4` interface that **MUST** be set by the server with the received message.

Return Values: The method **MUST** return `S_OK` (0x00000000) on success, or an implementation specific error `HRESULT` on failure.

When processing this call the server **MUST** follow these guidelines:

- If the *IsInitialized* instance variable equals `False`:
 - Return an error `OLE_E_BLANK` (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals `True`:
 - Return an error `MQ_ERROR_INVALID_HANDLE` (0xC00E0007) and take no further action.
- If the [Access Mode] of the [Open Queue] identified by the *OpenQueueReference* instance variable is not equal to `MQ_PEEK_ACCESS` or (`MQ_PEEK_ACCESS` | `MQ_ADMIN_ACCESS`) or `MQ_RECEIVE_ACCESS` or (`MQ_RECEIVE_ACCESS` | `MQ_ADMIN_ACCESS`):
 - Return an error `MQ_ERROR_ACCESS_DENIED` (0xC00E0025), and take no further action.
- If the *ppmsg* output parameter is `NULL`:
 - Return `E_INVALIDARG` (0x80070057) and take no further action.
- Define *suitable message* as the first [Message] from the tail of the [Messages Table] of the *referenced queue* for which the [IsLocked] property equals `False` or [AllowPeekWhenLocked] equals `True`.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the *ppmsg* output parameter to `NULL`.
 - Return `MQ_ERROR_MESSAGE_NOT_FOUND` (0xc00e0008) and take no further action.
 - Else, if there are one or more *suitable messages* available:
 - Retrieve the *suitable message*, and instantiate an `MSMQMessage` instance and initialize it with the *suitable message*, observing the requirements set forth by the *WantBody*, *WantDestinationQueue* and *WantConnectorType* input parameters. For details of initializing the `MSMQMessage` object, refer to section [3.17.3](#).
 - Set the *ppmsg* output parameter to the newly instantiated `MSMQMessage` instance.
- Return `S_OK` (0x00000000) and take no further action.

3.11.4.1.31 Purge (Opnum 37)

The **Purge** method is received by the server in an `RPC_REQUEST` packet. In response, the server Purges all [Message]s in the [Messages Table] of the *referenced queue*.

```
HRESULT Purge();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007) and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025) and take no further action.
- Remove all the [Message]s with [IsLocked] equals FALSE in the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.32 IsOpen2 (Opnum 38)

The **IsOpen2** method is received by the server in an RPC_REQUEST packet. In response, the server returns a value indicating whether the queue is open.

```
[propget] HRESULT IsOpen2(  
    [out, retval] VARIANT_BOOL* pisOpen  
);
```

pisOpen: A pointer to a VARIANT_BOOL that MUST be set to VARIANT_TRUE (0xffff) if the queue is open, or VARIANT_FALSE (0x0000) if the queue is closed.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST:

- If the *IsInitialized* instance variable equals False:
 - The server MUST return an error OLE_E_BLANK (0x80040007), and take no further action.
- If the *IsClosed* instance variable equals True:
 - Set the pisOpen output variable to VARIANT_FALSE (0x0000).
- Else:
 - Set the pisOpen output variable to VARIANT_TRUE (0xffff).
- Return S_OK (0x00000000) and take no further action.

3.11.4.1.33 ReceiveByLookupIdAllowPeek (Opnum 39)

The **ReceiveByLookupIdAllowPeek** method is received by the server in an RPC_REQUEST packet. In response, the server retrieves the [Message] from the *referenced queue's* [Messages Table] for which the [Lookup Identifier] property equals the lookup identifier, and removes it. If the receive is transactional, the message remains visible to peek operations.

```
HRESULT ReceiveByLookupIdAllowPeek(  
    [in] VARIANT LookupId,  
    [in, optional] VARIANT* Transaction,  
    [in, optional] VARIANT* WantDestinationQueue,  
    [in, optional] VARIANT* WantBody,  
    [in, optional] VARIANT* WantConnectorType,  
    [out, retval] MSMQMessage4** ppmsg  
);
```

LookupId: A VARIANT (VT_UI8) that contains a value that represents a position in the [Messages Table] of the *referenced queue* as described by the [Lookup Identifier] property of [Message].

Transaction: A pointer to a VARIANT that MUST contain either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an enlisted MSMQTransaction object.
- A VT_I4 that corresponds to one of the [MQTRANSACTION \(section 2.2.2.1\)](#) enumeration values.

If this parameter is not specified by the client, the server MUST use the default value MQ_MTS_TRANSACTION (0x00000001) in place of the unspecified value.

WantDestinationQueue: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the DestinationQueueInfo property set.
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the DestinationQueueInfo property set.

WantBody: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_TRUE (0xFFFF) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	Default. The server MUST return an MSMQMessage object that has the Body property set.
VARIANT_FALSE	The server MUST return an MSMQMessage object that does not have the Body

Value	Meaning
0x0000	property set.

WantConnectorType: A pointer to a VARIANT (VT_BOOL).

If this parameter is not specified by the client, the server MUST use the default value VARIANT_FALSE (0x0000) in place of the unspecified value.

Value	Meaning
VARIANT_TRUE 0xFFFF	The server MUST return an MSMQMessage object that has the ConnectorTypeGuid property set
VARIANT_FALSE 0x0000	Default. The server MUST return an MSMQMessage object that does not have the ConnectorTypeGuid property set.

ppmsg: A pointer to a pointer to an IMSMQMessage4 interface that MUST be set by the server with the received message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the LookupId input parameter equals 0:
 - Return error HRESULT and take no further action.
- If the *IsInitialized* instance variable equals False:
 - Return an error OLE_E_BLANK (0x80040007) and take no further action.
- If the *IsClosed* instance variable equals True:
 - Return an error MQ_ERROR_INVALID_HANDLE (0xC00E0007), and take no further action.
- If the [Access Mode] of the *OpenQueueReference* instance variable is not equal to MQ_RECEIVE_ACCESS or (MQ_RECEIVE_ACCESS | MQ_ADMIN_ACCESS):
 - Return an error MQ_ERROR_ACCESS_DENIED (0xC00E0025), and take no further action.
- If the ppmsg output parameter is NULL:
 - Return E_INVALIDARG (0x80070057) and take no further action.
- If the Transaction input parameter is not NULL:
 - If the Transaction input parameter is a VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object:
 - Define *transaction identifier* as the unique transaction identifier obtained from the ITransaction instance identified by the *Transaction* instance variable of the enlisted MSMQTransaction object.
 - Else, if the Transaction input parameter is a VT_I4:

- Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the Transaction input parameter.
- Else:
 - Return an error and take no further action.
- Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager], where the value of the [Transaction Identifier] property of the [Enlisted Transaction] equals the value of the transaction identifier.
- If an [Enlisted Transaction] cannot be located:
 - Create a new [Enlisted Transaction] and set the [Transaction Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] henceforth.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
- Define *suitable message* as the [Message] at the position in the [Messages Table] of the *referenced queue* identified by the LookupId input parameter (as described by the [Lookup Identifier] of [Message]), where the [IsLocked] property of the [Message] equals False.
- Attempt to retrieve the *suitable message*.
 - If no *suitable message* can be located:
 - Set the ppmsg output parameter to NULL.
 - Return MQ_ERROR_MESSAGE_NOT_FOUND (0xc00e0008) and take no further action.
 - Else:
 - Retrieve the *suitable message*, and instantiate an MSMQMessage instance and initialize it with the *suitable message*, observing the requirements set forth by the WantConnectorType, WantBody and WantDestinationQueue input parameters. For details of initializing the MSMQMessage object, refer to section [3.17.3](#).
 - Set the ppmsg output parameter to the newly instantiated MSMQMessage instance.
- If the Transaction input parameter is equal to MQ_MTS_TRANSACTION (0x00000001), MQ_XA_TRANSACTION (0x00000002), or a pointer to an MSMQTransaction instance:
 - Set the [IsLocked] property of the *suitable message* to True.
 - Set the [AllowPeekWhenLocked] property of the *suitable message* to True.
 - Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the *suitable message*.
 - The [Operation Type] property to the [Operation Type].Receive enumeration value.
 - Add the newly created [Transactional Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Else:

- If the *referenced queue* is an [Application Queue], and its [IsTargetJournalingEnabled] property is True:
 - Copy the *suitable message* to the [Messages Table] of the [Journal Queue] identified by the [Journal Queue] property of the *referenced queue*.
 - Remove the *suitable message* from the [Messages Table] of the *referenced queue*.
- Return S_OK (0x00000000) and take no further action.

3.11.5 Timer Events

There are no timer events for this coclass.

3.11.6 Other Local Events

There are no other local events for this coclass.

3.12 MSMQDestination Coclass

The MSMQDestination object identifies one or more [Queue]s, and enables the client to open and close them. The MSMQDestination object is used by other objects to send messages and to specify response queues.

An MSMQDestination object provides methods to identify one or more [Queue]s and to do the following tasks:

- Open an MSMQDestination object for sending messages to the identified [Queue]s.
- Close an MSMQDestination object and the corresponding [Open Queue].
- Check whether the MSMQDestination object is ready for use to send messages.

To identify [Queue]s and prepare an MSMQDestination object for use, one of the following methods MUST be invoked on an instance of this object prior to calling [MSMQDestination::Open](#):

- MSMQDestination::put_ADSPath, to specify the directory path that identifies [Queue]s.
- MSMQDestination::put_PathName, to specify the path name of one or more [Queue]s.
- MSMQDestination::put_FormatName, to specify the format name of one or more [Queue]s.

3.12.1 Abstract Data Model

An implementation of the [MSMQDestination](#) coclass MUST maintain the following abstract data elements:

- *QueueFormatName*: A string that represents the format name that identifies one or more [Queue]s.
- *QueueHandle*: A numeric value that matches the [Handle] property of an [Open Queue].
- *DirectoryPath*: A string that represents the directory path to one or more [Queue]s.
- *QueuePathName*: A string that represents the path name to one or more [Queue]s.

3.12.2 Timers

No protocol timers are required.

3.12.3 Initialization

When the object is constructed by a client, the server **MUST** set the instance variables as follows:

- Set the *QueueFormatName* instance variable to NULL.
- Set the *DirectoryPath* instance variable to NULL.
- Set the *QueuePathName* instance variable set to NULL.
- Set the *QueueHandle* instance variable to INVALID_HANDLE_VALUE (0xFFFFFFFF).

3.12.4 Message Processing Events and Sequencing Rules

The [MSMQDestination](#) coclass defines two interfaces: [IMSMQDestination](#) and [IMSMQPrivateDestination](#).

3.12.4.1 IMSMQDestination Interface

The [IMSMQDestination](#) interface provides methods for sending messages to queue(s). The version for this interface is 4.0.

To receive incoming remote calls for this interface, the server **MUST** implement a DCOM object class with the CLSID {eba96b18-2168-11d3-898c-00e02c074f6b} (coclass MSMQDestination as specified in section 1.9), which implements the IMSMQDestination interface using the UUID {eba96b16-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
Open (section 3.12.4.1.1)	Opens the MSMQDestination object for sending messages to queue(s) identified by the directory path, format name, or path name. Opnum: 7
Close (section 3.12.4.1.2)	Closes the open MSMQDestination object. Opnum: 8
IsOpen (section 3.12.4.1.3) , get IsOpen	Returns a Boolean indicating whether or not the MSMQDestination object is open for sending messages. Opnum: 9
IADs (section 3.12.4.1.4) , get IADs	The IMSMQDestination::get_IADs method is not implemented. Opnum: 10
IADs (section 3.12.4.1.5) , putref IADs	The IMSMQDestination::put_IADs method is not implemented. Opnum: 11

Method	Description
ADsPath (section 3.12.4.1.6) , get ADsPath	Returns the directory path that identifies the queue(s) represented by this object. Opnum: 12
ADsPath (section 3.12.4.1.7) , put ADsPath	Sets the directory path that identifies the queue(s) represented by this object. Opnum: 13
PathName (section 3.12.4.1.8) , get PathName	Returns the path name that identifies the queue(s) represented by this object. Opnum: 14
PathName (section 3.12.4.1.9) , put PathName	Sets the path name that identifies the queue represented by this object. Opnum: 15
FormatName (section 3.12.4.1.10) , get FormatName	Returns the format name that identifies the queue(s) represented by this object. Opnum: 16
FormatName (section 3.12.4.1.11) , put FormatName	Sets the format name that identifies the queue(s) represented by this object. Opnum: 17
Destinations (section 3.12.4.1.12) , get Destinations	The IMSMQDestination::get_Destinations method is not implemented. Opnum: 18
Destinations (section 3.12.4.1.13) , putref Destinations	The IMSMQDestination::put_Destinations method is not implemented. Opnum: 19
get Properties (section 3.12.4.1.14) , get Properties	The IMSMQDestination::get_Properties method is not implemented. Opnum: 20

3.12.4.1.1 Open (Opnum 7)

The **Open** method is received by the server in an RPC_REQUEST packet. In response, the server opens [Queue]s identified by the *QueueFormatName* instance variable for sending messages.

```
HRESULT Open () ;
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, S_FALSE (0x00000001) if the object is already opened, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *QueueHandle* instance variable is not equal to INVALID_HANDLE_VALUE (0xFFFFFFFF):
 - Return S_FALSE (0x00000001) and take no further action.

- If the *QueueFormatName* instance variable is NULL:
 - Return an error HRESULT and take no further action.
- Define *IsLocal* as a Boolean value that equals True if the *QueueFormatName* instance variable identifies a [Queue] that belongs to the [Queues Table] of the [Local Queue Manager]. Otherwise, *IsLocal* equals False.
- Define *IsPublic* as a Boolean value that equals True if the *QueueFormatName* instance variable identifies a [Public Queue]. Otherwise, *IsPublic* equals False.
- Define *SendingToOutgoing* as a Boolean value that MUST be set as follows:
 - If the *QueueFormatName* instance variable contains an HTTP or multicast format name, or identifies more than one queue, or *IsLocal* equals False:
 - Set *SendingToOutgoing* to True.
 - Else:
 - Set *SendingToOutgoing* to False.
- If *SendingToOutgoing* equals False:
 - Look up the [Queue] identified by the *QueueFormatName* instance variable, and term it as the *TargetQueue*.
 - If [Queue] doesn't exist:
 - Return an error HRESULT and take no further action.
- If *SendingToOutgoing* equals True:
 - Look up the [Outgoing Queue] in the [Queues Table] of the [Local Queue Manager] for which the [Destination Format Name] property equals *QueueFormatName*.
 - If [Outgoing Queue] is found, term it as the *TargetQueue*.
 - If [Outgoing Queue] doesn't exist:
 - Create an [Outgoing Queue], term it as the *TargetQueue*, and set the following property:
 - [Destination Format Name] to the value of the *QueueFormatName* instance variable.
 - Add the created [Outgoing Queue] to the [Queues Table] of the [Local Queue Manager].
 - Else:
 - Return an error HRESULT and take no further action.
- If there is no [Open Queue] whose [IsExclusive] property equals True in the [Open Queues Table] of the [Local Queue Manager] with a [Queue Reference] that identifies the *TargetQueue*:
 - Create an [Open Queue] that represents the *TargetQueue*, and set the following properties:
 - [IsExclusive] to False.
 - [Access] to MQ_SEND_ACCESS (0x00000002).

- [Queue Reference] to the *TargetQueue*.
- Add the created [Open Queue] to the [Opened Queues Table] of the [Local Queue Manager].
- Set the *QueueHandle* instance variable to a value of the [Handle] property of the created [Open Queue].
- Else:
 - Return an error HRESULT and take no further action.
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.2 Close (Opnum 8)

The **Close** method is received by the server in an RPC_REQUEST packet. In response, the server closes [Queue]s identified by the *QueueFormatName* instance variable.

```
HRESULT Close();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, S_FALSE (0x00000001) if the object is already closed, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *QueueHandle* instance variable equals INVALID_HANDLE_VALUE (0xFFFFFFFF):
 - Return S_FALSE (0x00000001) and take no further action.
- Remove the [Open Queue] identified by the *QueueHandle* instance variable from the [Open Queues Table] of the [Local Queue Manager].
- Set the *QueueHandle* instance variable to INVALID_HANDLE_VALUE (0xFFFFFFFF).
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.3 IsOpen (Opnum 9)

The **IsOpen** method is received by the server in an RPC_REQUEST packet. In response, the server returns a Boolean value indicating whether or not the [MSMQDestination](#) object is open for sending messages.

```
[propget] HRESULT IsOpen(
    [out, retval] VARIANT_BOOL* pfIsOpen
);
```

pfIsOpen: Pointer to a VARIANT_BOOL that, when set to VARIANT_TRUE (0xffff), indicates that the object is open. Otherwise, when set to VARIANT_FALSE (0x0000), it indicates that the object is closed.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *QueueHandle* instance variable equals `INVALID_HANDLE_VALUE` (0xFFFFFFFF):
 - Set the *pfIsOpen* output parameter to `VARIANT_FALSE` (0x00000000).
- Else:
 - Set the *pfIsOpen* output parameter to `VARIANT_TRUE` (0x00000001).
- Return `S_OK` (0x00000000) and take no further action.

3.12.4.1.4 IADs (Opnum 10)

The **IADs** method is not implemented.

```
[propget] HRESULT IADs(  
    [out, retval] IDispatch** ppIADs  
);
```

ppIADs: Pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return `E_NOTIMPL` (0x80004001).

When processing this call, the server MUST follow these guidelines:

- Return `E_NOTIMPL` (0x80004001) and take no further action.

3.12.4.1.5 IADs (Opnum 11)

The **IADs** method is not implemented.

```
[propputref] HRESULT IADs(  
    [in] IDispatch* pIADs  
);
```

pIADs: Pointer to an IDispatch interface. The server MUST ignore this parameter.

Return Values: The server MUST return `E_NOTIMPL` (0x80004001).

When processing this call, the server MUST follow these guidelines:

- Return `E_NOTIMPL` (0x80004001) and take no further action.

3.12.4.1.6 ADsPath (Opnum 12)

The **ADsPath** method is received by the server in an `RPC_REQUEST` packet. In response, the server returns the *DirectoryPath* instance variable that contains the directory path that identifies the [Queue]s represented by this object.

```
[propget] HRESULT ADsPath(  
    [out, retval] BSTR* pbstrADsPath  
);
```


pbstrADsPath: A pointer to a BSTR that contains the directory path that identifies the [Queue]s represented by this object.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Set the pbstrADsPath output parameter to the value of the *DirectoryPath* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.7 ADsPath (Opnum 13)

The **ADsPath** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *DirectoryPath* instance variable that contains the directory path that identifies the [Queue]s that are represented by this object.

```
[propput] HRESULT ADsPath(  
    [in] BSTR bstrADsPath  
);
```

bstrADsPath: A BSTR that contains the directory path to a [Directory Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Set the *DirectoryPath* instance variable to the bstrADsPath input parameter.
- Set the *QueueFormatName* instance variable to the format name that identifies the [Public Queue] exposed by the [Directory Queue] that is identified by the bstrADsPath input parameter.
- Set the *QueuePathName* instance variable to NULL.
- If the *QueueHandle* instance variable is not equal to INVALID_HANDLE_VALUE (0xFFFFFFFF), reopen the object by:
 1. Calling the [MSMQDestination::Close](#) method
 2. Calling the [MSMQDestination::Open](#) method.
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.8 PathName (Opnum 14)

The **PathName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *QueuePathName* instance variable that contains the path name that identifies the [Queue]s that are represented by this object.

```
[propget] HRESULT PathName(  
    [out, retval] BSTR* pbstrPathName  
);
```

pbstrPathName: A pointer to a BSTR that contains the UNC or DNS path name of the queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Set the pbstrPathName output parameter to the value of the *QueuePathName* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.9 PathName (Opnum 15)

The **PathName** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *QueuePathName* instance variable that contains the path name that identifies the [Queue]s that are represented by this object.

```
[propput] HRESULT PathName(  
    [in] BSTR bstrPathName  
);
```

bstrPathName: A BSTR that contains the UNC or DNS path name of the queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Set the *QueuePathName* instance variable to the bstrPathName input parameter.
- Set the *QueueFormatName* instance variable to the format name corresponding to the bstrPathName input parameter.
- Set the *DirectoryPath* instance variable to NULL.
- If the *QueueHandle* instance variable is not equal to INVALID_HANDLE_VALUE (0xFFFFFFFF), reopen the object by:
 1. Calling the [MSMQDestination::Close](#) method
 2. Calling the [MSMQDestination::Open](#) method.
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.10 FormatName (Opnum 16)

The **FormatName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the *QueueFormatName* instance variable that contains the format name that identifies the [Queue]s represented by this object.

```
[propget] HRESULT FormatName(  
    [out, retval] BSTR* pbstrFormatName  
);
```

pbstrFormatName: A pointer to a BSTR that contains the format name of the queue(s) represented by the [MSMQDestination](#) object.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- Set pbstrFormatName output parameter to the value of the *QueueFormatName* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.11 FormatName (Opnum 17)

The **FormatName** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *QueueFormatName* instance variable that contains the format name that identifies the [Queue]s represented by this object.

```
[propput] HRESULT FormatName(  
    [in] BSTR bstrFormatName  
);
```

bstrFormatName: A BSTR that contains the format name of the queue or queues represented by the [MSMQDestination](#) object.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- Set QueueFormatName instance variable to bstrFormatName input parameter.
- Set QueuePathName instance variable to NULL.
- Set DirectoryPath instance variable to NULL.
- If QueueHandle instance variable is not equal to INVALID_HANDLE_VALUE (0xFFFFFFFF), Reopen the object by:
 1. Calling the [MSMQDestination::Close](#) method
 2. Calling the [MSMQDestination::Open](#) method.
- Return S_OK (0x00000000) and take no further action.

3.12.4.1.12 Destinations (Opnum 18)

The **Destinations** method is not implemented.

```
[propget] HRESULT Destinations(  
    [out, retval] IDispatch** ppDestinations  
);
```

ppDestinations: Pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The **Destinations** method MUST take no action, and immediately return E_NOTIMPL (0x80004001).

3.12.4.1.13 Destinations (Opnum 19)

The **Destinations** method is not implemented.

```
[propputref] HRESULT Destinations(  
    [in] IDispatch* pDestinations  
);
```

pDestinations: Pointer to an IDispatch interface. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The **Destinations** method MUST take no action, and immediately return E_NOTIMPL (0x80004001).

3.12.4.1.14 Properties (Opnum 20)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(  
    [out, retval] IDispatch** ppcolProperties  
);
```

ppcolProperties: Pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

The Properties method MUST take no action and immediately return E_NOTIMPL (0x80004001).

3.12.4.2 IMSMQPrivateDestination Interface

The **IMSMQPrivateDestination** interface provides methods that for internal use by other objects like MSMQMessage to get the handle to the identified queue(s). The version for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {eba96b18-2168-11d3-898c-00e02c074f6b} (coclass MSMQDestination as specified in section 1.9), which implements the IMSMQPrivateDestination interface using the UUID {eba96b17-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
Handle (section 3.12.4.2.1) , get Handle	Returns the handle of the open queue. Opnum: 7

Method	Description
Handle (section 3.12.4.2.2) , put handle	Sets the handle of the open queue. Opnum: 8

3.12.4.2.1 Handle (Opnum 7)

The **Handle** method is received by the server in an RPC_REQUEST packet. In response, the server returns the value of the *QueueHandle* instance variable that identifies the [Open Queue].

```
[propget] HRESULT Handle(
    [out, retval] VARIANT* pvarHandle
);
```

pvarHandle: A pointer to a VARIANT that contains a VT_I8 value that identifies the [Open Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- Set pvarHandle output variable to the value of the *QueueHandle* instance variable.
- Return S_OK (0x00000000) and take no further action.

3.12.4.2.2 Handle (Opnum 8)

The **Handle** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *QueueHandle* instance variable that identifies the [Open Queue]. This method is internally used by other MSMQ objects like [MSMQMessage](#).

```
[propput] HRESULT Handle(
    [in] VARIANT varHandle
);
```

varHandle: VARIANT that contains a VT_I8 value that identifies the [Open Queue].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- Set *QueueHandle* instance variable to the value of the varHandle input parameter.
- Set *QueuePathName* instance variable to NULL.
- Set *DirectoryPath* instance variable to NULL.
- If varHandle input parameter value equals NULL or INVALID_HANDLE_VALUE
 - Set *QueueFormatName* instance variable to NULL.
- Else

- Set *QueueFormatName* instance variable to the format name that identifies the [Queue] identified by the [Queue Reference] property of the [Open Queue] that is identified by the *varHandle* input parameter.
- Return *S_OK* (0x00000000) and take no further action.

3.12.5 Timer Events

There are no timer events for this coclass.

3.12.6 Other Local Events

There are no other local events for this coclass.

3.13 MSMQQuery Coclass

The MSMQQuery coclass is used to create queries that are executed against the [Directory] to locate zero or more [Directory Queue]s that match a specified set of constraints. An MSMQQuery object is used to create and instantiate an [MSMQQueueInfos](#) object.

3.13.1 Abstract Data Model

No abstract data model is required.

3.13.2 Timers

No protocol timers are required.

3.13.3 Initialization

No initialization required.

3.13.4 Message Processing Events and Sequencing Rules

This coclass includes four interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQQuery4	IMSMQQuery3	IMSMQQuery2	IMSMQQuery
7	LookupQueue_v2	X	X	X	X
8	get Properties	X	X	X	
9	LookupQueue	X	X		

The interfaces represent revisions to the [MSMQQuery](#) class over time. IMSMQQuery is the oldest interface, and [IMSMQQuery4](#) is the newest.

3.13.4.1 IMSMQQuery4 Interface

The **IMSMQQuery4** interface provides methods for sending messages to queue(s). The version for this interface is 4.0.

There are 3 previous versions of this interface, IMSMQQuery, IMSMQQuery2 and IMSMQQuery3. These previous versions are nearly identical with a few less methods. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e073-dccd-11d0-aa4b-0060970debae} (coclass MSMQQuery as specified in section 1.9), which implements the IMSMQQuery4 interface using the UUID {eba96b24-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
LookupQueue_v2 (section 3.13.4.1.1)	Creates and returns an MSMQQueueInfos object, which defines a query that can be used to locate public queues. .<86> Opnum: 7
Properties (section 3.13.4.1.2), get Properties	Not implemented. .<87> Opnum: 8
LookupQueue (section 3.13.4.1.3)	Creates and returns an MSMQQueueInfos object, which defines a query that can be used to locate public queues. This method supersedes LookupQueue_v2, and includes additional query parameters. .<88> .<89> Opnum: 9

3.13.4.1.1 LookupQueue_v2 (Opnum 7)

The **LookupQueue_v2** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to an [IMSMQQueueInfos4](#) interface pointer. The returned interface allows the client to enumerate over a collection of [Directory Queue]s. The queue query represents search criteria based on any combination of [Queue Identifier GUID], [Service Type GUID], [Label], [Creation Time], or [Modification Time] properties of the [Directory Queue].

```
HRESULT LookupQueue_v2(  
    [in, optional] VARIANT* QueueGuid,  
    [in, optional] VARIANT* ServiceTypeGuid,  
    [in, optional] VARIANT* Label,  
    [in, optional] VARIANT* CreateTime,  
    [in, optional] VARIANT* ModifyTime,  
    [in, optional] VARIANT* RelServiceType,  
    [in, optional] VARIANT* RelLabel,  
    [in, optional] VARIANT* RelCreateTime,  
    [in, optional] VARIANT* RelModifyTime,  
    [out, retval] IMSMQQueueInfos4** ppqinfos  
);
```

QueueGuid: Pointer to a VARIANT containing a BSTR that contains a string representation of a [Queue Identifier GUID] property of the [Directory Queue]. The string MUST adhere to the following format, specified using Augmented BNF:

```
guid = "{" dword-part "-" word-part "-" word-part "-"  
        2byte-part "-" 6byte-part "}"  
dword-part = 2word-part  
word-part = 2byte-part  
byte-part = 2hex-digit  
hex-digit = %x30-39 / %x41-46 / %x61-66
```

ServiceTypeGuid: Pointer to a VARIANT containing a BSTR that contains a string representation of a [Service Type GUID] property of the [Directory Queue]. The string MUST adhere to the same format defined for the QueueGuid input parameter.

Label: Pointer to a VARIANT containing a BSTR that is used to query against the [Label] property of a [Directory Queue].

CreateTime: Pointer to a VARIANT that contains a DATE value that is used to query against the [Creation Time] property of a [Directory Queue].

ModifyTime: Pointer to a VARIANT that contains a DATE value that is used to query against the [Modification Time] property of the [Directory Queue].

RelServiceType: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the ServiceTypeGuid input parameter. This parameter corresponds to the [RELOPS \(section 2.2.2.20\)](#) enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

RelLabel: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the Label input parameter. This parameter corresponds to the **RELOPS** enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

RelCreateTime: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the CreateTime input parameter. This parameter corresponds to the **RELOPS** enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

RelModifyTime: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the ModifyTime input parameter. This parameter corresponds to the **RELOPS** enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

ppqinfos: Pointer to an **IMSMQQueueInfos4** interface pointer, that upon successful completion contains a pointer to an [MSMQQueueInfos](#) instance. This MSMQQueueInfos instance contains the query definition that can be used by the client to enumerate a collection of [Directory Queue]s that match the specified criteria.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create and initialize a new MSMQQueueInfos instance object, and pass the values of the following input parameters to the initialization of the MSMQQueueInfos instance

object. For information on the initialization process, refer to section [3.14.3](#). These values are referred to as query constraints.

- QueueGuid
- ServiceTypeGuid
- Label
- CreateTime
- ModifyTime
- RelServiceType
- RelLabel
- RelCreateTime
- RelModifyTime
- The query constraints MUST be combined using a logical AND operator when querying for matching queues.
- The unspecified (VT_ERROR) input parameters MUST not be passed as query constraints to the initialization of the new MSMQQueueInfos object.
- The server MUST initialize the MSMQQueueInfos object by supplying the appropriate constraints, as shown above, prior to returning the interface pointer to the client.
- If initialization of the MSMQQueueInfos object instance succeeds:
 - The server MUST set the ppqinfos output parameter to a pointer to an **IMSMQQueueInfos4** interface pointer for the newly created MSMQQueueInfos object.
- Else:
 - The server MUST return an error HRESULT, and take no further action.

3.13.4.1.2 Properties (Opnum 8)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(  
    [out, retval] IDispatch** ppcolProperties  
);
```

ppcolProperties: Pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

When processing this call the server MUST follow these guidelines:

- The server MUST return E_NOTIMPL and take no further action.(0x80004001).

3.13.4.1.3 LookupQueue (Opnum 9)

The **LookupQueue** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to an [IMSMQQueueInfos4](#) interface pointer. The returned interface allows the client to enumerate over a collection of [Directory Queue]s. The queue query represents search criteria based on any combination of [Queue Identifier GUID], [Service Type GUID], [Label], [Creation Time], [Modification Time], or [Multicast Address] properties of the [Directory Queue].

```
HRESULT LookupQueue (
    [in, optional] VARIANT* QueueGuid,
    [in, optional] VARIANT* ServiceTypeGuid,
    [in, optional] VARIANT* Label,
    [in, optional] VARIANT* CreateTime,
    [in, optional] VARIANT* ModifyTime,
    [in, optional] VARIANT* RelServiceType,
    [in, optional] VARIANT* RelLabel,
    [in, optional] VARIANT* RelCreateTime,
    [in, optional] VARIANT* RelModifyTime,
    [in, optional] VARIANT* MulticastAddress,
    [in, optional] VARIANT* RelMulticastAddress,
    [out, retval] IMSMQQueueInfos4** ppqinfos
);
```

QueueGuid: Pointer to a VARIANT containing a BSTR that contains a string representation of a [Queue Identifier GUID] property of the [Directory Queue]. The string MUST adhere to the following format, specified using Augmented BNF:

```
guid = "{" dword-part "-" word-part "-" word-part "-"
        2byte-part "-" 6byte-part "}"
dword-part = 2word-part
word-part = 2byte-part
byte-part = 2hex-digit
hex-digit = %x30-39 / %x41-46 / %x61-66
```

ServiceTypeGuid: Pointer to a VARIANT containing a BSTR that contains a string representation of a [Service Type GUID] property of the [Directory Queue]. The string MUST adhere to the same format defined for QueueGuid.

Label: Pointer to a VARIANT containing a BSTR that is used to query against the [Label] property of a [Directory Queue].

CreateTime: Pointer to a VARIANT that contains a DATE that is used to query against the [Creation Time] property of a [Directory Queue].

ModifyTime: Pointer to a VARIANT that contains a DATE that is used to query against the [Modification Time] property of a [Directory Queue].

RelServiceType: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the ServiceTypeGuid input parameter. This parameter corresponds to the [RELOPS \(section 2.2.2.20\)](#) enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

RelLabel: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the Label input parameter. This parameter corresponds to the **RELOPS**

enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

RelCreateTime: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the CreateTime input parameter. This parameter corresponds to the **RELOPS** enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

RelModifyTime: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the ModifyTime input parameter. This parameter corresponds to the **RELOPS** enum. If not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

MulticastAddress: Pointer to a VARIANT that contains a BSTR that is used to query against the [Multicast Address] property of a [Directory Queue]. If this value is not specified, or an empty string is passed in, the server MUST create a query restriction that matches all queues that do not have a [Multicast Address] property defined.

RelMulticastAddress: Pointer to a VARIANT that contains a VT_I4 that specifies a logical comparison operator against the MulticastAddress input parameter. This parameter corresponds to the **RELOPS** enum. If this value is not specified by the client, the server MUST use the default value REL_EQ (0x00000001) in place of the unspecified value.

ppqinfos: Pointer to an **IMSMQQueueInfos4** interface pointer, that upon successful completion contains a pointer to a [MSMQQueueInfos](#) instance containing the query definition that can be used by the client to enumerate a collection of [Directory Queue]s that match the specified criteria.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create and initialize a new MSMQQueueInfos object, and pass the values of the following input parameters to the initialization of the MSMQQueueInfos object. For information on the initialization process, refer to section [3.14.3](#). These values are referred to as query constraints.
 - QueueGuid
 - ServiceTypeGuid
 - Label
 - CreateTime
 - ModifyTime
 - RelServiceType
 - RelLabel
 - RelCreateTime
 - RelModifyTime
 - MulticastAddress

- RelMulticastAddress
- The query constraints MUST be combined using a logical AND operator when querying for matching queues.
- The unspecified (VT_ERROR) input parameters MUST not be passed as query constraints to the initialization of the new MSMQQueueInfos object.
- The server MUST initialize the MSMQQueueInfos object by supplying the appropriate constraints, as shown above, prior to returning the interface pointer to the client.
- If initialization of the MSMQQueueInfos object instance succeeds:
 - The server MUST set the ppqinfos output parameter to a pointer to an **IMSMQQueueInfos4** interface pointer for the newly created MSMQQueueInfos object.
- Else:
 - The server MUST return an error HRESULT, and take no further action.

3.13.5 Timer Events

No timer events are required.

3.13.6 Other Local Events

No local events are required.

3.14 MSMQQueueInfos Coclass

The MSMQQueueInfos coclass is created by an [MSMQQuery](#) object, and is used by a client application to query for and enumerate over a collection of [Directory Queue]s from the [Directory]. An MSMQQueueInfos object dynamically queries the [Directory], and therefore the query results collection might change if new [Directory Queue]s are added that match the query constraints, or if an existing [Directory Queue]'s properties are modified so that they no longer match the specified query constraints.

3.14.1 Abstract Data Model

The [MSMQQueueInfos](#) object MUST maintain the following data elements:

- *QueueCollection*: A collection of [Directory Queues] from the [Directory Queues Table] in the [Directory] that match the specified search criteria.
- *CollectionCursor*: A cursor that tracks the current position within the *QueueCollection* instance variable when enumerating over the results.
- *IsInitialized*: A value that tracks the initialized state of the object.

3.14.2 Timers

No protocol timers are required.

3.14.3 Initialization

The server MUST provide an instance of this object to the client in an initialized state as a result of [MSMQQuery::LookupQueue v2 \(section 3.13.4.1.1\)](#) or [MSMQQuery::LookupQueue \(section 3.13.4.1.3\)](#) calls. To initialize the object, the query constraints MUST be set by the [MSMQQuery](#) object that created this instance.

- The initialized query constraints MUST be used to populate the *QueueCollection* instance variable. The server MUST populate the *QueueCollection* instance variable by querying all the [Directory Queues Table]s in the [Directory], applying the specified query constraints and adding every matching [Directory Queue]. The query constraints MUST be combined using a logical AND operator.
- Once initialization is complete, the server MUST set the *IsInitialized* instance variable to True.

If a new instance of the object is created by the client, the *IsInitialized* instance variable MUST be set to False and the rest of the instance variables set to NULL.

3.14.4 Message Processing Events and Sequencing Rules

This coclass includes four interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQQueueInfos4	IMSMQQueueInfos3	IMSMQQueueInfos2	IMSMQQueueInfos
7	Reset	X	X	X	X
8	Next	X	X	X	X
9	get Properties	X	X	X	

3.14.4.1 IMSMQQueueInfos4 Interface

The **IMSMQQueueInfos4** interface provides methods for interacting with a collection of queue(s). The version for this interface is 4.0.

There are 3 previous versions of this interface, *IMSMQQueueInfos*, *IMSMQQueueInfos2* and *IMSMQQueueInfos3*. These previous versions are nearly identical with a few less methods. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e07e-dccd-11d0-aa4b-0060970debae} (coclass *MSMQQueueInfos* as specified in section 1.9), which implements the *IMSMQQueueInfos4* interface using the UUID {eba96b22-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section **3.1.4**.

Methods in RPC Opnum Order

Method	Description
Reset (section 3.14.4.1.1)	Moves the <i>CollectionCursor</i> instance variable to the head of the <i>QueueCollection</i> instance variable. Opnum: 7
Next (section 3.14.4.1.2)	Returns an <i>MSMQQueueInfo</i> object that represents the next [Directory Queue] in the <i>QueueCollection</i> instance variable. <90> Opnum: 8
Properties (section 3.14.4.1.3) , get Properties	This method is not implemented. <91> Opnum: 9

3.14.4.1.1 Reset (Opnum 7)

The **Reset** method is received by the server in an RPC_REQUEST packet. In response, the server resets the *CollectionCursor* instance variable to the head of the *QueueCollection* instance variable.

```
HRESULT Reset(
    void
);
```

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *IsInitialized* instance variable equals False:
 - The server MUST return an error HRESULT and take no further action.
- The server MUST set the *CollectionCursor* instance variable to point to the head of the collection represented by the *QueueCollection* instance variable.

3.14.4.1.2 Next (Opnum 8)

The **Next** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [MSMQQueueInfo4](#) object that represents a [Public Queue] exposed by the next [Directory Queue] in the *QueueCollection*.

```
Next(
    [out, retval] IMSMQQueueInfo4** ppqinfoNext
);
```

ppqinfoNext: A pointer to an **MSMQQueueInfo4** interface pointer that upon successful completion contains an initialized [MSMQQueueInfo](#) object representing the [Public Queue] that

is exposed by the next [Directory Queue] within the collection represented by the *QueueCollection* instance variable.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the IsInitialized instance variable equals False
 - The server MUST return an error HRESULT.
- If the CollectionCursor instance variable is at the end of the collection represented by the QueueCollection instance variable
 - The server MUST set the ppqinfoNext output parameter to NULL and take no further action.
- Else
 - Create a new MSMQQueueInfo object instance
 - Set the QueueFormatName instance variable of the created MSMQQueueInfo object to the format name identifying the [Public Queue] that is exposed by the [Directory Queue] identified by the CollectionCursor instance variable in the collection represented by QueueCollection instance variable.
 - Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.
 - Set the ppqinfoNext output parameter to a pointer to an IMSMQQueryInfo4 interface pointer of the newly created MSMQQueueInfo object.
- The server MUST advance the cursor represented by the CollectionCursor instance variable to the next [Directory Queue] in the collection represented by the QueueCollection instance variable.

3.14.4.1.3 Properties (Opnum 9)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(  
    [out, retval] IDispatch** ppcolProperties  
);
```

ppcolProperties: Pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001);

When processing this call the server MUST follow these guidelines:

- The Properties method MUST return E_NOTIMPL (0x80004001) and take no further action.

3.14.5 Timer Events

No timer events are required.

3.14.6 Other Local Events

No local events are required.

3.15 MSMQCollection Coclass

The MSMQCollection coclass encapsulates a collection of VARIANT values, and is used by [MSMQOutgoingQueueManagement](#) and [MSMQQueueManagement](#) to provide sets of property values to the client.

3.15.1 Abstract Data Model

The [MSMQCollection](#) object MUST maintain the following data elements:

VariantCollection: A read-only collection of VARIANT objects stored by key/value pair association that MUST allow for random access to elements. The key MUST be a BSTR containing the name, and the value MUST be a VARIANT object. The collection contents are established at initialization time.

3.15.2 Timers

No protocol timers are required.

3.15.3 Initialization

- It is expected that the [MSMQCollection](#) object is created and the *VariantCollection* instance variable is populated internally by calls to either [IMSMQOutgoingQueueManagment::EodGetSendInfo \(section 3.5.4.1.3\)](#) or [IMSMQQueueManagment::EodGetReceiveInfo \(section 3.4.4.1.3\)](#).
- If an instance of MSMQCollection is created by any means other than **IMSMQOutgoingQueueManagment::EodGetSendInfo** or **IMSMQQueueManagment::EodGetReceiveInfo**, the *VariantCollection* instance variable MUST be initialized to an empty collection.

3.15.4 Message Processing Events and Sequencing Rules

The [MSMQCollection](#) coclass defines a single interface: [IMSMQCollection](#).

3.15.4.1 IMSMQCollection Interface

The **IMSMQCollection** interface provides methods for interacting with a VariantCollection of queue(s). The version for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {f72b9031-2f0c-43e8-924e-e6052cdc493f} (coclass MSMQCollection as specified in section 1.9), which implements the IMSMQCollection interface using the UUID {0188ac2f-ecb3-4173-9779-635ca2039c72}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section 3.2.1.5.8. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section 3.1.4.

Methods in RPC Opnum Order

Method	Description
Item (section 3.15.4.1.1)	Returns the specific element from the <i>VariantCollection</i> instance variable with the specified key value. Opnum: 7
Count (section 3.15.4.1.2) , get Count	Returns the number of elements in the <i>VariantCollection</i> instance variable. Opnum: 8
NewEnum (section 3.15.4.1.3)	Returns a standard OLE Automation enumerator as defined by [MS-OAUT] section 2.2.20 that can be used to enumerate the <i>VariantCollection</i> instance variable. Opnum: 9

3.15.4.1.1 Item (Opnum 7)

The **Item** method is received by the server in an RPC_REQUEST packet. In response, the server returns a VARIANT from the *VariantCollection* instance variable, where the key matches with the Index input parameter.

```
HRESULT Item(
    [in] VARIANT* Index,
    [out, retval] VARIANT* pvarRet
);
```

Index: A pointer to a VARIANT containing a BSTR which contains the key value used to look up the associated VARIANT value in the *VariantCollection* instance variable.

pvarRet: A pointer to a VARIANT that upon success will contain the element returned from the *VariantCollection* instance variable.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the *VariantCollection* instance variable does not contain an element where the key value matches the Index input parameter
 - The server MUST return MQ_ERROR_INVALID_PARAMETER (0xC00E0006) and take no further action
- The pvarRet output parameter MUST be set to the associated VARIANT in the *VariantCollection* instance variable where the key value matches the Index input parameter.

3.15.4.1.2 Count (Opnum 8)

The **Count** method is received by the server in an RPC_REQUEST packet. In response, the server returns the total number of elements in the *VariantCollection* instance variable.

```
[propget] HRESULT Count(
    [out, retval] long* pCount
);
```

pCount: A pointer to a LONG that contains the number of elements in the *VariantCollection* instance variable.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- The pCount output parameter MUST be set to the number of elements in the *VariantCollection* instance variable.

3.15.4.1.3 _NewEnum (Opnum 9)

The **_NewEnum** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to an IUnknown interface pointer that represents an enumerator instance for this collection. **_NewEnum** implements the IEnumVARIANT Server functionality as described in [MS-OAUT] section 3.3

```
[restricted] HRESULT _NewEnum(  
    [out, retval] IUnknown** ppunk  
);
```

ppunk: The ppunk output parameter MUST be set to an IUnknown pointer to a standard OLE automation enumerator that can be used to enumerate the *VariantCollection* instance variable elements.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- The ppunk output parameter MUST be set to an IUnknown pointer to a standard OLE automation enumerator that can be used to enumerate the *VariantCollection* instance variable elements.

3.15.5 Timer Events

No timer events are required.

3.15.6 Other Local Events

No local events are required.

3.16 MSMQEvent Coclass

The MSMQEvent object provides an event notification facility that a client can configure to receive asynchronous notifications regarding the availability of [Message]s in a specific [Queue]. The relationship between the [MSMQQueue](#) object instance representing the specific [Queue] and the MSMQEvent is established through the [MSMQQueue4::EnableNotification](#) method. The MSMQEvent class MUST support the IConnectionPoint and IConnectionPointContainer interfaces through which client callbacks are implemented. Each client that registers with an MSMQEvent object instance MUST provide a valid pointer to an object that implements the [DMSMQEventEvents](#) interface. For further information regarding the IConnectionPoint and IConnectionPointContainer interfaces, refer to [Box98].

3.16.1 Abstract Data Model

The [MSMQEvent](#) object MUST maintain the following data elements:

- *ConnectionCollection*

A collection of [_DMSMQEventEvents \(section 3.16.4.3\)](#) interface pointers that are implemented by the client applications. These pointers can be used to callback to the client to notify them that an event has been raised. The collection MUST be stored as a key/value pair association. The key for each item is an integer value, referred to here as a *ConnectionCookie*, that uniquely identifies each item within the *ConnectionCollection* instance variable. The value associated with each key is a **_DMSMQEventEvents** interface pointer.

- *ConnectionLimit*

An integer value representing the limit on the number of **_DMSMQEventEvents** interface pointers that can be registered. This value defines the maximum number of entries stored in the *ConnectionCollection* instance variable.

3.16.2 Timers

No protocol timers are required.

3.16.3 Initialization

The *ConnectionCollection* instance variable MUST be initialized to an empty collection.

3.16.4 Message Processing Events and Sequencing Rules

This coclass defines three numbered interfaces that are binary-compatible revisions. The numbered interfaces may append additional methods, and/or update method parameter types. The following table illustrates the methods belonging to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQEvent3	IMSMQEvent2	IMSMQEvent
7	get Properties	X	X	

Additionally, the server MUST implement the standard DCOM event interfaces: [IConnectionPoint](#) and [IConnectionPointContainer](#). The server MAY implement the [IMSMQPrivateEvent](#) interface.

This coclass also defines an outgoing event interface: [_DMSMQEventEvents](#).

3.16.4.1 IMSMQEvent3 Interface

The **IMSMQEvent3** interface provides methods for sending messages to queue(s). The version for this interface is 4.0.

There are 2 previous versions of this interface, IMSMQEvent and IMSMQEvent2. These previous versions are nearly identical with a few less methods. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e07a-dccd-11d0-aa4b-0060970debae} (coclass MSMQEvent as specified in

section [1.9](#)), which implements the IMSMQEvent3 interface using the UUID {eba96b1c-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section **3.1.4**.

Methods in RPC Opnum Order

Method	Description
Properties (section 3.16.4.1.1) , get Properties	This method is not implemented. <92> Opnum: 7

3.16.4.1.1 Properties (Opnum 7)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(  
    [out, retval] IDispatch** ppcolProperties  
);
```

ppcolProperties: A pointer to an IDispatch pointer.

Return Values: The server MUST return E_NOTIMPL (0x80004001).

When processing this call the server MUST follow these guidelines:

- The Properties method MUST return E_NOTIMPL (0x80004001) and take no further action.

3.16.4.2 IMSMQPrivateEvent Interface

The **IMSMQPrivateEvent** interface provides methods for internal communication between the MSMQEvent and the MSMQQueue to which it was registered through the MSMQQueue::EnableNotification method. The version for this interface is 4.0.

This interface is not required for client/server communication and as such is optional. [<93>](#)

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e07a-dccd-11d0-aa4b-0060970debae} (coclass MSMQEvent as specified in section [1.9](#)), which implements the IMSMQPrivateEvent interface using the UUID {d7ab3341-c9d3-11d1-bb47-0080c7c5a2c0}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section **3.1.4**.

Methods in RPC Opnum Order

Method	Description
Hwnd (section 3.16.4.2.1) , get Hwnd	Returns a pointer to a LONG . Opnum: 7

Method	Description
FireArrivedEvent (section 3.16.4.2.2)	Provides notification of the arrival of a [Message]. Opnum: 8
FireArrivedErrorEvent (section 3.16.4.2.3)	Provides notification of an arrival error, such as the expiry of the receive timeout. Opnum: 9

3.16.4.2.1 Hwnd (Opnum 7)

The **Hwnd** method is received by the server in an RPC_REQUEST packet. In response, the server returns a **LONG** value which the client MUST ignore. Since the returned **LONG** value serves no purpose, the server MAY [<94>](#) return 0x00000000.

```
[propget] HRESULT Hwnd(
    [out, retval] long* phwnd
);
```

phwnd: A pointer to a **LONG** that MAY contain 0x00000000. The value returned via this parameter MUST be ignored by the client, and serves no purpose.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

This interface is optional for communication with the client. If implemented, this interface MUST reside on the server.

3.16.4.2.2 FireArrivedEvent (Opnum 8)

The **FireArrivedEvent** method is received by the server in an RPC_REQUEST packet. In response, the server provides notification of the availability of a [Message].

```
HRESULT FireArrivedEvent(
    [in] IMSMQQueue* pq,
    [in] long msgcursor
);
```

pq: A pointer to an [IMSMQQueue](#) interface that upon success will be cast to an IDispatch pointer.

msgcursor: A LONG value that specifies the value of the cursor option that was specified through the Cursor input parameter that was passed to the [IMSMQQueue4::EnableNotification](#) operation to associate this [MSMQEvent](#) with the [MSMQQueue](#). This parameter corresponds to the [MQMSGCURSOR \(section 2.2.2.8\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

3.16.4.2.3 FireArrivedErrorEvent (Opnum 9)

The **FireArrivedErrorEvent** method is received by the server in an RPC_REQUEST packet. In response, the server provides notification of an error relating to the arrival of a message.

```
HRESULT FireArrivedErrorEvent(  
    [in] IMSMQQueue* pq,  
    [in] HRESULT hrStatus,  
    [in] long msgcursor  
);
```

pq: A pointer to an [IMSMQQueue](#) interface, that upon success will be cast to an IDispatch pointer and forwarded on to the client.

hrStatus: An HRESULT value that specifies the error code that was received from the [Queue] where the [Message] was delivered.

msgcursor: A LONG value that specifies the value of the cursor option that was specified through the Cursor input parameter that was passed to the [IMSMQQueue4::EnableNotification](#) operation to associate this [MSMQEvent](#) with the queue. This parameter corresponds to the [MQMSGCURSOR \(section 2.2.2.8\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

3.16.4.3 _DMSMQEventEvents Interface

The **_DMSMQEventEvents Interface** is an outgoing interface that MUST be implemented by a client in order to receive event notifications. The version for this interface is 4.0.

To receive incoming remote calls for this interface, the client MUST implement a DCOM interface using the UUID {D7D6E078-DCCD-11d0-AA4B-0060970DEBAE}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [\[MS-OAUT\]](#) section **3.1.4**.

Methods in RPC Opnum Order

Method	Description
Arrived (section 3.16.4.2.1)	Called by an MSMQEvent object to signal that a message has arrived. Opnum: 7
ArrivedError (section 3.16.4.2.2)	Called by an MSMQEvent object to signal that an error has occurred. Opnum: 8

3.16.4.3.1 Arrived (Opnum 7)

The **Arrived** method is implemented by a client and is invoked by the server to notify a message arrival event.

```
void Arrived(
    [in] IDispatch* Queue,
    [in] long Cursor
);
```

Queue: A pointer to an IDispatch interface for the [Queue] that raised the event.

Cursor: A LONG value that specifies the value of the cursor option that was specified through the Cursor input parameter passed to the [IMSMQQueue4::EnableNotification](#) operation to associate this MSMQEvent with the queue. This parameter corresponds to the [MQMSGCOURSE](#) enum as defined in section [2.2.2.8](#).

This method has no return values.

3.16.4.3.2 ArrivedError (Opnum 8)

The **ArrivedError** method is implemented by a client and is invoked by the server to notify a message arrival error event.

```
void ArrivedError(
    [in] IDispatch* Queue,
    [in] long ErrorCode,
    [in] long Cursor
);
```

Queue: A pointer to an IDispatch interface for the [MSMQQueue](#) that is associated with the event.

ErrorCode: A LONG value that specifies the error code. The error code is an HRESULT casted to LONG. [<95>](#)

Cursor: A LONG value that specifies the value of the cursor option that was specified through the Cursor input parameter passed to the [IMSMQQueue4::EnableNotification](#) operation to associate this MSMQEvent with the queue. This parameter corresponds to the [MQMSGCOURSE](#) enum as defined in section [2.2.2.8](#).

This method has no return values.

3.16.4.4 IConnectionPoint Interface

The **IConnectionPoint** interface provides methods for interacting with connection points. The version for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM interface using the UUID {B196B286-BAB4-101A-B69C-00AA00341D07}.

The opnum table below begins at opnum 3. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**.

Methods in RPC Opnum Order

Method	Description
GetConnectionInterface (section	Returns the interface identifier (IID) of the interface that client

Method	Description
3.16.4.4.1	applications MUST implement for the MSMQEvent object to be able to notify them when an event is raised. Opnum: 3
GetConnectionPointContainer (section 3.16.4.4.2)	Returns the pointer to an IConnectionPointContainer interface for the MSMQEvent object. Opnum: 4
Advise (section 3.16.4.4.3)	Registers a client's callback object to receive event notifications. Opnum: 5
Unadvise (section 3.16.4.4.4)	Deregisters a client's callback object to cease receiving event notifications. Opnum: 6
EnumConnections (section 3.16.4.4.5)	Returns a standard OLE enumerator as defined by [MS-OAUT] section 2.2.20 that can be used to enumerate the currently registered callback objects. Opnum: 7

3.16.4.4.1 GetConnectionInterface (Opnum 3)

The **GetConnectionInterface** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to the interface identifier (IID) of the interface that client applications MUST implement in order for the [MSMQEvent](#) object to be able to notify them when an event is raised.

```
HRESULT GetConnectionInterface(
    [out] IID* pIID
);
```

pIID: A pointer to an interface identifier (IID), that upon successful completion will contain the IID value of the [_DMSMQEventEvents](#) interface.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<96>](#)

When processing this call, the server MUST follow these guidelines:

- If the piid output parameter equals NULL:
 - The server MUST return E_POINTER (0x80000005).
- Else:
 - The server MUST set the piid output parameter to the IID for the [_DMSMQEventEvents](#) interface (D7D6E078-DCCD-11d0-AA4B-0060970DEBAE)

3.16.4.4.2 GetConnectionPointContainer (Opnum 4)

The **GetConnectionPointContainer** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to a pointer to an [IConnectionPointContainer](#) interface for the [MSMQEvent](#) object.


```
HRESULT GetConnectionPointContainer(
    [out] IConnectionPointContainer** ppCPC
);
```

ppCPC: A pointer to an **IConnectionPointContainer** interface pointer for the MSMQEvent object.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<97>](#)

When processing this, call the server MUST follow these guidelines:

- If the ppCPC output parameter equals NULL:
 - The server MUST return E_POINTER (0x80000005).
- Else:
 - The server MUST set the piid output parameter to a pointer to an IConnectionPointContainer interface for the MSMQEvent object.

3.16.4.4.3 Advise (Opnum 5)

The **Advise** method is received by the server in an RPC_REQUEST packet. In response, the server registers a client's callback object to receive event notifications.

```
HRESULT Advise(
    [in] IUnknown* pUnkSink,
    [out] DWORD* pdwCookie
);
```

pUnkSink: A pointer to an IUnknown (as specified in [MS-DCOM](#) section **3.2.1.5.8**) interface for the client object.

pdwCookie: A pointer to a unique DWORD value that uniquely identifies the new element in the *ConnectionCollection* instance variable.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<98>](#)

When processing this call, the server MUST follow these guidelines:

- If the pUnkSink input parameter equals NULL:
 - The server MUST return E_POINTER (0x80000005), and take no further action.
- If the pdwCookie output parameter equals NULL:
 - The server MUST return E_POINTER (0x80000005), and take no further action.
- If the number of elements in the *ConnectionCollection* instance variable is equal to the *ConnectionLimit* instance variable.
 - The server MUST return CONNECT_E_ADVISELIMIT (0x80040201), and take no further action.

- The server MUST attempt to obtain a pointer to the [DMSMQEventEvents](#) interface by calling IUnknown::QueryInterface (refer to section [3.1](#)) on the pUnkSink input parameter.
- If the IUnknown::QueryInterface method returns an error:
 - The server MUST set the pdwCookies value to zero (0x00000000).
 - The server MUST return CONNECT_E_CANNOTCONNECT (0x80040202), and take no further action.
- The server MUST set the value of the pdwCookie pointer output parameter to a DWORD "cookie" that is a unique key within the ConnectionCollection instance variable.
- The server MUST add a new item to the *ConnectionCollection* instance variable, using the value of the pdwCookie as the *ConnectionCookie*, and the interface pointer acquired above as the associated value.

3.16.4.4.4 Unadvise (Opnum 6)

The **Unadvise** method is received by the server in an RPC_REQUEST packet. In response, the server deregisters a client's callback object to cease receiving event notifications.

```
HRESULT Unadvise(
    [in] DWORD dwCookie
);
```

dwCookie: A DWORD value that uniquely identifies the interface pointer for the callback that is to be deregistered. This corresponds to the value of the "cookie" that was returned in the call to [IConnectionPoint::Advise](#).

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<99>](#)

When processing this call, the server MUST follow these guidelines:

- If the dwCookie input parameter value does NOT exist as a key in the *ConnectionCollection* collection:
 - The server MUST return CONNECT_E_NOCONNECTION (0x80040200) and take no further action.
- The server MUST call the IUnknown::Release method (refer to section [3.1](#)) on the interface pointer that is retrieved from the *ConnectionCollection* instance variable where the key value matches the dwCookie input parameter.
- The server MUST remove the item from the *ConnectionCollection* instance variable where the key value matches the dwCookie input parameter.

3.16.4.4.5 EnumConnections (Opnum 7)

The **EnumConnections** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to an IEnumConnections interface pointer for the client to enumerate all the currently registered callback objects.

```
HRESULT EnumConnections(
```

```
[out] IEnumConnections** ppEnum
);
```

ppEnum: A pointer to an IEnumConnections interface pointer, that upon successful completion will allow the user to enumerate all the currently registered callback objects.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<100>](#)

When processing this call, the server MUST follow these guidelines:

- If the ppEnum output parameter equals NULL:
 - The server MUST return E_POINTER (0x80000005), and take no further action.
- If the server does not implement this method, it MUST return E_NOTIMPL (0x80004001) and take no further action.
- The server MUST set the ppEnum output parameter to a pointer to an IEnumConnections interface for the [MSMQEvent](#) object instance. The ppEnum output parameter will enable enumeration of all the currently registered callback objects that exist in the *ConnectionCollection* instance variable.

3.16.4.5 IConnectionPointContainer Interface

The **IConnectionPointContainer** interface provides methods for enumerating the connection points within a container. The version for this interface is 4.0.

To receive incoming remote calls for this interface, the server MUST implement a DCOM interface using the UUID {B196B284-BAB4-101A-B69C-00AA00341D07}.

The opnum table below begins at opnum 3. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [\[MS-DCOM\]](#) section **3.2.1.5.8**.

Methods in RPC Opnum Order

Method	Description
EnumConnectionPoints (section 3.16.4.5.1)	Returns a pointer to a pointer for an IEnumConnections interface that can be used by the client to enumerate all the IConnectionPoint implementations for the MSMQEvent object. Opnum: 3
FindConnectionPoint (section 3.16.4.5.2)	Returns a pointer to an IConnectionPoint interface for a specified interface identifier (IID). Opnum: 4

3.16.4.5.1 EnumConnectionPoints (Opnum 3)

The **EnumConnectionPoints** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to an IEnumConnectionPoints interface pointer that can be used by the client to enumerate all the IConnectionPoint implementations for the [MSMQEvent](#) object.

```
HRESULT EnumConnectionPoints(
    [out] IEnumConnectionPoints** ppEnum
);
```

ppEnum: A pointer to an IEnumConnectionPoints interface pointer, that upon successful completion will allow the user to enumerate all the IConnectionPoint implementations for the MSMQEvent object.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<101>](#)

When processing this call, the server MUST follow these guidelines:

- If the ppEnum output parameter equals NULL:
 - The server MUST return E_POINTER (0x80000005), and take no further action.
- The server MUST set the ppEnum output parameter to a pointer to an IEnumConnectionPoints interface for the MSMQEvent object.

3.16.4.5.2 FindConnectionPoint (Opnum 4)

The **FindConnectionPoint** method is received by the server in an RPC_REQUEST packet. In response, the server returns a pointer to an IConnectionPoint interface pointer for a specified interface identifier (IID).

```
HRESULT FindConnectionPoint(
    [in] REFIID riid,
    [out] IConnectionPoint** ppCP
);
```

riid: The IID of the interface whose connection point object is being requested.

ppCP: A pointer to a pointer to an IConnectionPoint interface, that supports the interface identified by the IID in the riid input parameter.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure. [<102>](#)

When processing this call, the server MUST follow these guidelines:

- If the ppCP output parameter equals NULL
 - The server MUST return E_POINTER (0x80000005), and take no further action.
- If the riid input parameter is NOT equal to the IID of the [_DMSMQEventEvents](#) interface ({D7D6E078-DCCD-11d0-AA4B-0060970DEBAE}):
 - The server MUST return CONNECT_E_NOCONNECTION (0x80040200), and take no further action.
- Else
 - The server MUST set the ppCP output parameter to a pointer to an IConnectionPoint interface for the [MSMQEvent](#) object.

3.16.5 Timer Events

No timer events are required.

3.16.6 Other Local Events

No local events are required.

3.17 MSMQMessage Coclass

The MSMQMessage coclass encapsulates a single instance of a [Message]. It provides methods for setting and retrieving all [Message] properties, as well as functionality for sending the [Message] to an [Application Queue].

3.17.1 Abstract Data Model

The [MSMQMessage](#) class represents a single instance of a [Message]. For specific details on the data elements refer to section [3.1.1.23](#).

The MSMQMessage object MUST maintain the following data elements:

- *Message*
The [Message] being represented by this object instance.
- *RequestedAuthenticationLevel*
An instance state variable that holds the requested authentication level from the client. This variable corresponds to the [MQMSGAUTHLEVEL \(section 2.2.2.16\)](#) enumeration.
- *SOAPBody*
An instance state variable, of type BSTR, that holds a string value containing extra SOAP elements that can be specified by the client application to be appended to the body of an SRMP Message.
- *SOAPHeader*
An instance state variable, of type BSTR, that holds a string value containing extra SOAP elements that can be specified by the client application to be appended to the header of an SRMP Message
- *SenderIdIdentifierCache*
An instance state variable that is of the same type as the [Sender Identifier] property of a [Message]. This allows the user to cache the value of the [Sender Identifier] property for optimization when sending multiple [Message]s with the same [Sender Identifier] property value. It is set with the [AttachCurrentSecurityContext \(section 3.17.4.1.49\)](#) and [AttachCurrentSecurityContext2 \(section 3.17.4.1.87\)](#) functions. This instance variable persists its value for the lifetime of the MSMQMessage object, and once it is set, it can only be altered by subsequent calls to the **AttachCurrentSecurityContext** and **AttachCurrentSecurityContext2** functions.
- *SenderCertificateCache*
An instance state variable that is of the same type as the [Sender Certificate] property of a [Message]. This allows the user to cache the value of the [Sender Certificate] property for

optimization when sending multiple [Message]s with the same [Sender Certificate] property value. It is set with the **AttachCurrentSecurityContext** (section 3.17.4.1.49) and **AttachCurrentSecurityContext2** (section 3.17.4.1.87) functions. This instance variable persists its value for the lifetime of the MSMQMessage object, and once it is set, it can only be altered by subsequent calls to the **AttachCurrentSecurityContext** (section 3.17.4.1.49) and **AttachCurrentSecurityContext2** (section 3.17.4.1.87) functions.

- *SymmetricKeyCache*

An instance state variable that is identical to the [Symmetric Key] property of a [Message]. This allows the user to cache the value of the [Symmetric Key] property for optimization when sending multiple [Message]s with the same [Symmetric Key] property value. It is set with the **AttachCurrentSecurityContext** (section 3.17.4.1.49) and **AttachCurrentSecurityContext2** (section 3.17.4.1.87) functions. This instance variable persists its value for the lifetime of the MSMQMessage object, and once it is set, it can only be altered by subsequent calls to the **AttachCurrentSecurityContext** (section 3.17.4.1.49) and **AttachCurrentSecurityContext2** (section 3.17.4.1.87) functions.

3.17.2 Timers

No protocol timers are required.

3.17.3 Initialization

An [MSMQMessage](#) object can be initialized in two different ways: when it is created by the client application for sending, or when it is created by the server as a result of receiving the message. When a message is received, unless otherwise stated, the initialized values for the MSMQMessage object are populated with the corresponding values in the [Message] that was retrieved from the [Message Table]. The default values when an MSMQMessage object is created by the client are shown in the table below. For specific details on the data elements, refer to section [3.1.1.23](#).

Data element	Initial value – client created	Initial value – server created
[Lookup Identifier]	(0x00000000)	
[Message Identifier]	NULL	
[Class]	Invalid Value< 103 >	
[Privacy Level]	MQMSG_PRIV_LEVEL_NONE (0x00000000)	
[Authentication Level]	MQMSG_AUTHENTICATION_NOT_REQUESTED (0x00000000)	
[Delivery Guarantee]	MQMSG_DELIVERY_EXPRESS (0x00000000)	
[Tracing Requested]	MQMSG_TRACING_NONE (0x00000000)	

Data element	Initial value – client created	Initial value – server created
[Priority]	(0x00000003)	
[Journaling Requested]	MQMSG_JOURNAL_NONE (0x00000000)	
[Response Queue Format Name]	NULL	
[AppSpecific]	NULL	
[Source Machine GUID]	GUID_NULL {00000000-0000-0000-0000-000000000000}	
[Body]	NULL	
[Administration Queue Format Name]	NULL	
[Correlation Identifier]	NULL	
[Acknowledgements Requested]	MQMSG_ACKNOWLEDGEMENT_NONE (0x00000000)	
[Label]	NULL	
[Time To Reach Queue]	(0xFFFFFFFF)	
[Time To Be Received]	(0xFFFFFFFF)	
[Hash Algorithm]	MQMSG_CALG_SHA1 (0x00008004)	
[Encryption Algorithm]	MQMSG_CALG_RC4 (0x00000801)	
[Sent Time]	(0xFFFFFFFF)	
[Arrived Time]	(0xFFFFFFFF)	
[Destination Queue Format Name]	NULL	

Data element	Initial value – client created	Initial value – server created
[Sender Certificate]	NULL	
[Sender Identifier Type]	MQMSG_SENDERID_TYPE_SID (0x00000001)	
[Sender Identifier]	NULL	
[Extension]	NULL	
[Connector Type GUID]	GUID_NULL {00000000-0000-0000-0000-000000000000}	
[Transaction Status Queue Format Name]	NULL	
[Symmetric Key]	NULL	
[Signature]	NULL	
[Authentication Provider Type]	(0x00000001)	
[Authentication Provider Name]	"Microsoft Base Cryptographic Provider, v1,0"	
[Transactional Message Sequence Identifier]	NULL	
[IsFirstInTransaction]	False	
[IsLastInTransaction]	False	
[SOAP Envelope]	NULL	
[SOAP Compound Message]	NULL	
RequestedAuthenticationLevel	MQMSG_AUTH_LEVEL_NONE (0x00000000)	MQMSG_AUTH_LEVEL_NONE (0x00000000)
SOAPBody	NULL	NULL

Data element	Initial value – client created	Initial value – server created
SOAPHeader	NULL	NULL
SenderIdCache	NULL	NULL
SenderCertificateCache	NULL	NULL
SymmetricKeyCache	NULL	NULL

3.17.4 Message Processing Events and Sequencing Rules

This coclass includes four interfaces. The numbered interfaces are binary-compatible revisions that may append additional methods, and/or update method parameter types. The following table illustrates the methods that belong to each interface revision.

Opnum	Method name (in the most recent interface revision)	IMSMQMessa ge4	IMSMQMessa ge3	IMSMQMessa ge2	IMSMQMess age
7	get Class	X	X	X	X
8	get PrivLevel	X	X	X	X
9	put PrivLevel	X	X	X	X
10	get AuthLevel	X	X	X	X
11	put AuthLevel	X	X	X	X
12	get IsAuthenticated	X	X	X	X
13	get Delivery	X	X	X	X
14	put Delivery	X	X	X	X
15	get Trace	X	X	X	X
16	put Trace	X	X	X	X
17	get Priority	X	X	X	X
18	put Priority	X	X	X	X
19	get Journal	X	X	X	X
20	put Journal	X	X	X	X
21	ResponseQueueInfo v1	X	X	X	X
22	putref ResponseQueueInfo v1	X	X	X	X
23	get AppSpecific	X	X	X	X
24	put AppSpecific	X	X	X	X

Opnum	Method name (in the most recent interface revision)	IMSMQMessage4	IMSMQMessage3	IMSMQMessage2	IMSMQMessage
25	get SourceMachineGuid	X	X	X	X
26	get BodyLength	X	X	X	X
27	get Body	X	X	X	X
28	put Body	X	X	X	X
29	AdminQueueInfo_v1	X	X	X	X
30	putref AdminQueueInfo_v1	X	X	X	X
31	get Id	X	X	X	X
32	get CorrelationId	X	X	X	X
33	put CorrelationId	X	X	X	X
34	get Ack	X	X	X	X
35	put Ack	X	X	X	X
36	get Label	X	X	X	X
37	put Label	X	X	X	X
38	get MaxTimeToReachQueue	X	X	X	X
39	put MaxTimeToReachQueue	X	X	X	X
40	get MaxTimeToReceive	X	X	X	X
41	put MaxTimeToReceive	X	X	X	X
42	get HashAlgorithm	X	X	X	X
43	put HashAlgorithm	X	X	X	X
44	get EncryptAlgorithm	X	X	X	X
45	put EncryptAlgorithm	X	X	X	X
46	get SentTime	X	X	X	X
47	get ArrivedTime	X	X	X	X
48	get DestinationQueueInfo	X	X	X	X
49	get SenderCertificate	X	X	X	X
50	put SenderCertificate	X	X	X	X

Opnum	Method name (in the most recent interface revision)	IMSMQMessage4	IMSMQMessage3	IMSMQMessage2	IMSMQMessage
51	get SenderId	X	X	X	X
52	get SenderIdType	X	X	X	X
53	put SenderIdType	X	X	X	X
54	Send	X	X	X	X
55	AttachCurrentSecurityContext	X	X	X	X
56	get SenderVersion	X	X	X	
57	get Extension	X	X	X	
58	put Extension	X	X	X	
59	get ConnectorTypeGuid	X	X	X	
60	put ConnectorTypeGuid	X	X	X	
61	get TransactionStatusQueueInfo	X	X	X	
62	get DestinationSymmetricKey	X	X	X	
63	put DestinationSymmetricKey	X	X	X	
64	get Signature	X	X	X	
65	put Signature	X	X	X	
66	get AuthenticationProviderType	X	X	X	
67	put AuthenticationProviderType	X	X	X	

Opnum	Method name (in the most recent interface revision)	IMSMQMessage4	IMSMQMessage3	IMSMQMessage2	IMSMQMessage
68	get AuthenticationProviderName	X	X	X	
69	put AuthenticationProviderName	X	X	X	
70	put SenderId	X	X	X	
71	get MsgClass	X	X	X	
72	put MsgClass	X	X	X	
73	get Properties	X	X	X	
74	get TransactionId	X	X	X	
75	get IsFirstInTransaction	X	X	X	
76	get IsLastInTransaction	X	X	X	
77	ResponseQueueInfo_v2	X	X	X	
78	ResponseQueueInfo_v2	X	X	X	
79	AdminQueueInfo_v2	X	X	X	
80	AdminQueueInfo_v2	X	X	X	
81	get ReceivedAuthenticationLevel	X	X	X	
82	get ResponseQueueInfo	X	X		
83	putref ResponseQueueInfo	X	X		

Opnum	Method name (in the most recent interface revision)	IMSMQMessage4	IMSMQMessage3	IMSMQMessage2	IMSMQMessage
84	get AdminQueueInfo	X	X		
85	putref AdminQueueInfo	X	X		
86	get ResponseDestination	X	X		
87	putref ResponseDestination	X	X		
88	get Destination	X	X		
89	get LookupId	X	X		
90	get IsAuthenticated2	X	X		
91	get IsFirstInTransaction2	X	X		
92	get IsLastInTransaction2	X	X		
93	AttachCurrentSecurityContext2	X	X		
94	get SoapEnvelope	X	X		
95	get CompoundMessage	X	X		
96	put SoapHeader	X	X		
97	put SoapBody	X	X		

3.17.4.1 IMSMQMessage4 Interface

The **IMSMQMessage4** interface provides methods for sending messages to queue(s). The version for this interface is 4.0.

There are 3 previous versions of this interface, IMSMQMessage, IMSMQMessage2 and IMSMQMessage3. These previous versions are nearly identical with a few less methods. All differences from previous versions are described in Windows Behavior notes in the method descriptions that follow.

To receive incoming remote calls for this interface, the server MUST implement a DCOM object class with the CLSID {d7d6e075-dccd-11d0-aa4b-0060970debae} (coclass MSMQMessage as specified in section 1.9), which implements the IMSMQMessage4 interface using the UUID {eba96b23-2168-11d3-898c-00e02c074f6b}.

The opnum table below begins at opnum 7. Opnums 0 through 2 are inherited from the IUnknown interface as specified in [MS-DCOM] section 3.2.1.5.8. Opnums 3 through 6 are inherited from the IDispatch interface as specified in [MS-OAUT] section 3.1.4.

Methods in RPC Opnum Order

Method	Description
Class (section 3.17.4.1.1) , get Class	Returns the message type Opnum: 7
PrivLevel (section 3.17.4.1.2) , get PrivLevel	Returns the privacy level of the message Opnum: 8
PrivLevel (section 3.17.4.1.3) , put PrivLevel	Sets the privacy level of the message Opnum: 9
AuthLevel (section 3.17.4.1.4) , get AuthLevel	Returns the authentication level of the message Opnum: 10
AuthLevel (section 3.17.4.1.5) , put AuthLevel	Sets the authentication level of the message Opnum: 11
IsAuthenticated (section 3.17.4.1.6) , get IsAuthenticated	Returns a Boolean indicating whether the message was authenticated at the request of the sending application Opnum: 12
Delivery (section 3.17.4.1.7) , get Delivery	Returns the message delivery guarantee Opnum: 13
Delivery (section 3.17.4.1.8) , put Delivery	Sets the message delivery guarantee Opnum: 14
Trace (section 3.17.4.1.9) , get Trace	Returns the requested level of tracing__Description_Goes_Here__ Opnum: 15
Trace (section 3.17.4.1.10) , put Trace	Sets the requested level of tracing Opnum: 16
Priority (section 3.17.4.1.11) , get Priority	Returns the message priority Opnum: 17
Priority (section 3.17.4.1.12) , put Priority	Sets the message priority Opnum: 18
Journal (section 3.17.4.1.13) , get Journal	Returns the requested journaling setting Opnum: 19
Journal (section 3.17.4.1.14) , put	Sets the requested journaling setting

Method	Description
Journal	Opnum: 20
ResponseQueueInfo v1 (section 3.17.4.1.15)	Returns an MSMQQueueInfo object that represents the [Queue] that will receive response messages from the receiving application Opnum: 21
ResponseQueueInfo v1 (section 3.17.4.1.16) , putref ResponseQueueInfo v1	Sets the queue that will receive response messages from the receiving application. The client MUST pass an MSMQQueueInfo object that represents the [Queue] Opnum: 22
AppSpecific (section 3.17.4.1.17) , get AppSpecific	Returns an application-specific value Opnum: 23
AppSpecific (section 3.17.4.1.18) , put AppSpecific	Sets an application-specific value Opnum: 24
SourceMachineGuid (section 3.17.4.1.19) , get SourceMachineGuid	Returns the GUID identifier of the source computer that sent the message Opnum: 25
BodyLength (section 3.17.4.1.20) , get BodyLength	Returns the length of the message body Opnum: 26
Body (section 3.17.4.1.21) , get Body	Returns the body of the message Opnum: 27
Body (section 3.17.4.1.22) , put Body	Sets the body of the message Opnum: 28
AdminQueueInfo v1 (section 3.17.4.1.23)	Returns the [Queue] that will receive system generated acknowledgement and error messages. Opnum: 29
AdminQueueInfo v1 (section 3.17.4.1.24) , putref AdminQueueInfo v1	Sets the [Queue] that will receive system-generated acknowledgement and error messages. Opnum: 30
Id (section 3.17.4.1.25) , get Id	Returns the unique system-generated message identifier Opnum: 31
CorrelationId (section 3.17.4.1.26) , get CorrelationId	Returns the application-specific correlation identifier Opnum: 32
CorrelationId (section 3.17.4.1.27) , put CorrelationId	Sets the application-specific correlation identifier Opnum: 33
Ack (section 3.17.4.1.28) , get Ack	Returns the type of acknowledgement message that will be posted for the message Opnum: 34
Ack (section 3.17.4.1.29) , put Ack	Sets the type of acknowledgement message that will be posted for the message

Method	Description
	Opnum: 35
Label (section 3.17.4.1.30) , get Label	Returns the application-specific description of the message Opnum: 36
Label (section 3.17.4.1.31) , put Label	Sets the application-specific description of the message Opnum: 37
MaxTimeToReachQueue (section 3.17.4.1.32) , get MaxTimeToReachQueue	Returns the maximum number of seconds for the message to reach the queue. Opnum: 38
MaxTimeToReachQueue (section 3.17.4.1.33) , put MaxTimeToReachQueue	Sets the maximum number of seconds for the message to reach the queue. Opnum: 39
MaxTimeToReceive (section 3.17.4.1.34) , get MaxTimeToReceive	Returns the maximum number of seconds for the message to be received from the queue Opnum: 40
MaxTimeToReceive (section 3.17.4.1.35) , put MaxTimeToReceive	Sets the maximum number of seconds for the message to be received from the queue Opnum: 41
HashAlgorithm (section 3.17.4.1.36) , get HashAlgorithm	Returns the hashing algorithm used when authenticating the message Opnum: 42
HashAlgorithm (section 3.17.4.1.37) , put HashAlgorithm	Sets the hashing algorithm used when authenticating the message Opnum: 43
EncryptAlgorithm (section 3.17.4.1.38) , get EncryptionAlgorithm	Returns the encryption algorithm used to encrypt the body of a message when encryption is requested Opnum: 44
EncryptAlgorithm (section 3.17.4.1.39) , put EncryptionAlgorithm	Sets the encryption algorithm used to encrypt the body of a message when encryption is requested Opnum: 45
SentTime (section 3.17.4.1.40) , get SentTime	Returns the UTC time that the message was sent Opnum: 46
ArrivedTime (section 3.17.4.1.41) , get ArrivedTime	Returns the UTC time that the message arrived in the [Queue] Opnum: 47
DestinationQueueInfo (section 3.17.4.1.42) , get DestinationQueueInfo	Returns the [Queue] to which the message is addressed. <104> Opnum: 48
SenderCertificate (section 3.17.4.1.43) , get SenderCertificate	Returns the user certificate that is used to authenticate the message Opnum: 49

Method	Description
SenderCertificate (section 3.17.4.1.44) , put SenderCertificate	Sets the user certificate that is used to authenticate the message Opnum: 50
SenderId (section 3.17.4.1.45) , get SenderId	Returns the identifier of the sending user Opnum: 51
SenderIdType (section 3.17.4.1.46) , get SenderType	Returns the type of sender identifier used Opnum: 52
SenderIdType (section 3.17.4.1.47) , put SenderIdType	Sets the type of sender identifier used Opnum: 53
Send (section 3.17.4.1.48)	Sends the message Opnum: 54
AttachCurrentSecurityContext (section 3.17.4.1.49)	Stores a cached copy of all the information required to attach a certificate to a message for authentication. If the values are not specified it will query the [Directory] for the values. Opnum: 55
SenderVersion (section 3.17.4.1.50) , get SenderVersion	Returns the version of transfer used to send the message.<105> Opnum: 56
Extension (section 3.17.4.1.51) , get Extension	Returns application-specific information that is associated with the message.<106> Opnum: 57
Extension (section 3.17.4.1.52) , put Extension	Sets application-specific information that is associated with the message.<107> Opnum: 58
ConnectorTypeGuid (section 3.17.4.1.53) , get ConnectorTypeGuid	Returns an application-specific GUID that is used to identify the connector application.<108> Opnum: 59
ConnectorTypeGuid (section 3.17.4.1.54) , put ConnectorTypeGuid	Sets an application-specific GUID that is used to identify the connector application.<109> Opnum: 60
TransactionStatusQueueInfo (section 3.17.4.1.55) , get TransactionStatusQueueInfo	Returns the [Queue] to which connector applications will send acknowledgement messages when sending to a foreign transactional queue.<110><111> Opnum: 61
DestinationSymmetricKey (section 3.17.4.1.56) , get DestinationSymetricKey	Returns the symmetric key used to encrypt the message.<112> Opnum: 62
DestinationSymmetricKey (section 3.17.4.1.57) , put DestinationSymmetricKey	Sets the symmetric key used to encrypt the message.<113>

Method	Description
	Opnum: 63
Signature (section 3.17.4.1.58) , get Signature	Returns the digital signature that is attached to the message. <114> Opnum: 64
Signature (section 3.17.4.1.59) , put Signature	Sets the digital signature that is attached to the message. <115> Opnum: 65
AuthenticationProviderType (section 3.17.4.1.60) , get AuthenticationProviderType	Returns the type of cryptographic provider used to generate the digital signature that is attached to the message. <116> Opnum: 66
AuthenticationProviderType (section 3.17.4.1.61) , put AuthenticationProviderType	Sets the type of cryptographic provider used to generate the digital signature that is attached to the message. <117> Opnum: 67
AuthenticationProviderName (section 3.17.4.1.62) , get AuthenticationProviderName	Returns the name of the cryptographic provider used to generate the digital signature that is attached to the message. <118> Opnum: 68
AuthenticationProviderName (section 3.17.4.1.63) , put AuthenticationProviderName	Sets the name of the cryptographic provider used to generate the digital signature that is attached to the message. <119> Opnum: 69
SenderId (section 3.17.4.1.64) , put SenderId	Sets the identifier of the sending user. <120> Opnum: 70
MsgClass (section 3.17.4.1.65) , get MsgClass	Returns the message type. <121> Opnum: 71
MsgClass (section 3.17.4.1.66) , put MsgClass	Sets the message type. <122> Opnum: 72
Properties (section 3.17.4.1.67) , get Properties	This method is not implemented. <123> Opnum: 73
TransactionId (section 3.17.4.1.68) , get TransactionId	Returns the identifier of the transaction within the scope of which this message was sent. <124> Opnum: 74
IsFirstInTransaction (section 3.17.4.1.69) , get IsFirstInTransaction	Returns an indicator of whether the message was the first message sent in the transaction. <125> Opnum: 75
IsLastInTransaction (section 3.17.4.1.70) , get IsLastInTransaction	Returns an indicator of whether the message was the last message sent in the transaction. <126> Opnum: 76

Method	Description
ResponseQueueInfo v2 (section 3.17.4.1.71)	Returns the [Queue] that will receive response messages from the receiving application. .<127> Opnum: 77
ResponseQueueInfo v2 (section 3.17.4.1.72)	Sets the [Queue] that will receive response messages from the receiving application. .<128> Opnum: 78
AdminQueueInfo v2 (section 3.17.4.1.73)	Returns the [Queue] that will receive system-generated acknowledgement messages. .<129> Opnum: 79
AdminQueueInfo v2 (section 3.17.4.1.74)	Sets the [Queue] that will receive system-generated acknowledgement messages. .<130> Opnum: 80
ReceivedAuthenticationLevel (section 3.17.4.1.75) , get ReceivedAuthenticationLevel	Returns an indicator specifying if the message was authenticated and what digital signature was used. .<131> Opnum: 81
ResponseQueueInfo (section 3.17.4.1.76) , get ResponseQueueInfo	Returns the [Queue] that will receive response messages from the receiving application. .<132><133> Opnum: 82
ResponseQueueInfo (section 3.17.4.1.77) , putref ResponseQueueInfo	Sets the [Queue] that will receive response messages from the receiving application. .<134><135> Opnum: 83
AdminQueueInfo (section 3.17.4.1.78) , get AdminQueueInfo	Returns the [Queue] that will receive message queuing generated acknowledgement messages. .<136><137> Opnum: 84
AdminQueueInfo (section 3.17.4.1.79) , putref AdminQueueInfo	Sets the [Queue] that will receive message queuing generated acknowledgement messages. .<138><139> Opnum: 85
ResponseDestination (section 3.17.4.1.80) , get ResponseDestination	Returns the destination that represents 0 or more [Queue]s that will receive response messages from the receiving application. .<140> Opnum: 86
ResponseDestination (section 3.17.4.1.81) , putref ResponseDestination	Sets the destination that represents 0 or more [Queue]s that will receive response messages from the receiving application. .<141> Opnum: 87
Destination (section 3.17.4.1.82) , get Destination	Returns the destination that represents 0 or more [Queue]s that the [Message] will be sent to. .<142> Opnum: 88
LookupId (section 3.17.4.1.83) , get LookupId	Returns the lookup identifier for the message. .<143> Opnum: 89
IsAuthenticated2 (section 3.17.4.1.84)	Returns whether the message was authenticated at the

Method	Description
, get IsAuthenticated2	request of the sending application. .<144> Opnum: 90
IsFirstInTransaction2 (section 3.17.4.1.85) , get IsFirstInTransaction2	Returns an indicator of whether the message was the first message sent in the transaction. .<145> Opnum: 91
IsLastInTransaction2 (section 3.17.4.1.86) , get IsLastInTransaction2	Returns an indicator of whether the message was the last message sent in the transaction. .<146> Opnum: 92
AttachCurrentSecurityContext2 (section 3.17.4.1.87)	Stores a cached copy of all the information required to attach a certificate to a message for authentication. If the values are not specified it will query the [Directory] for the values. .<147> Opnum: 93
SoapEnvelope (section 3.17.4.1.88) , get SoapEnvelope	Returns the SOAP envelope attached to an SRMP message. .<148> Opnum: 94
CompoundMessage (section 3.17.4.1.89) , get CompoundMessage	Returns the entire SRMP message including the soap envelope and soap attachments. .<149> Opnum: 95
SoapHeader (section 3.17.4.1.90) , put SoapHeader	Sets additional application-specific header elements for inclusion in the soap envelope of an SRMP message. .<150> Opnum: 96
SoapBody (section 3.17.4.1.91) , put SoapBody	Sets additional application-specific body elements for inclusion in the soap envelope of an SRMP message. .<151> Opnum: 97

3.17.4.1.1 Class (Opnum 7)

The **Class** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Class] property of the represented [Message].

This method is deprecated; implementing this method is optional. [.<152>](#)

```
[propget] HRESULT Class(
    [out, retval] long* pClass
);
```

pClass: A pointer to a LONG that identifies the message type. This parameter corresponds to the [MQMSGCLASS \(section 2.2.2.9\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plClass output parameter MUST be set to the value of the [Class] property of the represented [Message].

3.17.4.1.2 PrivLevel (Opnum 8)

The **PrivLevel** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Privacy Level] property of the represented [Message].

```
[propget] HRESULT PrivLevel(
    [out, retval] long* plPrivLevel
);
```

plPrivLevel: A pointer to a long integer that identifies the privacy level of the message. This parameter corresponds to the [MQPRIVLEVEL \(section 2.2.2.7\)](#) enum. <153>

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plPrivLevel output parameter MUST be set to the value of the [Privacy Level] property of the represented [Message].

3.17.4.1.3 PrivLevel (Opnum 9)

The **PrivLevel** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Privacy Level] property of the represented [Message].

```
[propput] HRESULT PrivLevel(
    [in] long lPrivLevel
);
```

lPrivLevel: A long integer value that contains the privacy level to set. This parameter corresponds to the [MQMSGPRIVLEVEL \(section 2.2.2.15\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the lPrivLevel input parameter is NOT a valid **MQMSGPRIVLEVEL** enumeration value:
 - The server MUST return MQ_ERROR_ILLEGAL_PROPERTY_VALUE (0x 0xC00E0018L), and take no further action.
- The [Privacy Level] property of the represented [Message] MUST be assigned the value contained in the lPrivLevel input parameter.

3.17.4.1.4 AuthLevel (Opnum 10)

The **AuthLevel** method is received by the server in an RPC_REQUEST packet. In response, the server returns the value of the *RequestedAuthenticationLevel* instance variable.

```
[propget] HRESULT AuthLevel(
    [out, retval] long* plAuthLevel
);
```

plAuthLevel: A pointer to a long integer that identifies the authentication level of the message. This parameter corresponds to the [MQMSGAUTHLEVEL \(section 2.2.2.16\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plAuthLevel output parameter MUST be set to the value of the *RequestedAuthenticationLevel* instance variable.

3.17.4.1.5 AuthLevel (Opnum 11)

The **AuthLevel** method is received by the server in an RPC_REQUEST packet. In response, the server sets the value of the *RequestedAuthenticationLevel* instance variable.

```
[propput] HRESULT AuthLevel(
    [in] long lAuthLevel
);
```

lAuthLevel: A long integer that identifies the authentication level of the message. This parameter corresponds to the [MQMSGAUTHLEVEL \(section 2.2.2.16\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the lAuthLevel input parameter is NOT a valid MQMSGAUTHLEVEL enumeration value:
 - The server MUST return MQ_ERROR_ILLEGAL_PROPERTY_VALUE (0x 0xC00E0018), and take no further action.
- The *RequestedAuthenticationLevel* instance variable MUST be set to the value contained in the lAuthLevel input parameter.

3.17.4.1.6 IsAuthenticated (Opnum 12)

The **IsAuthenticated** method is received by the server in an RPC_REQUEST packet. In response, the server returns a Boolean flag indicating whether the message was authenticated by the [Queue Manager] that received the message.

```
[propget] HRESULT IsAuthenticated(
    [out, retval] short* pisAuthenticated
);
```

pisAuthenticated: A pointer to a short that specifies whether or not the message was authenticated.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the [Authentication Level] property of the represented [Message] is MQMSG_AUTHENTICATION_NOT_REQUESTED (0x00000000):
 - The pisAuthenticated output parameter MUST be set to 0x0000.
- Else:
 - The pisAuthenticated output parameter MUST be set to 0x0001.

3.17.4.1.7 Delivery (Opnum 13)

The **Delivery** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Delivery Guarantee] property of the represented [Message].

```
[propget] HRESULT Delivery(  
    [out, retval] long* plDelivery  
);
```

plDelivery: A pointer to a long integer that identifies the delivery method of the message. This parameter corresponds to the [MQMSGDELIVERY \(section 2.2.2.10\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plDelivery output parameter MUST be set to the value of the [Delivery Guarantee] property of the represented [Message].

3.17.4.1.8 Delivery (Opnum 14)

The **Delivery** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Delivery Guarantee] property of the represented [Message].

```
[propput] HRESULT Delivery(  
    [in] long lDelivery  
);
```

lDelivery: A long integer that identifies the delivery method of the message. This parameter corresponds to the [MQMSGDELIVERY \(section 2.2.2.10\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the lDelivery input parameter is NOT a valid **MQMSGDELIVERY** enumeration value:
 - The server MUST return MQ_ERROR_ILLEGAL_PROPERTY_VALUE (0x 0xC00E0018L), and take no further action

- The [Delivery Guarantee] property of the represented [Message] MUST be set to the value contained in the IDelivery input parameter.

3.17.4.1.9 Trace (Opnum 15)

The **Trace** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Tracing Requested] property of the represented [Message].

```
[propget] HRESULT Trace(
    [out, retval] long* plTrace
);
```

plTrace: A pointer to a long integer that identifies the trace method for the message. This parameter corresponds to the [MQMSGTRACE \(section 2.2.2.13\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plTrace output parameter MUST be set to the value of the [Tracing Requested] property of the represented [Message].

3.17.4.1.10 Trace (Opnum 16)

The **Trace** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Tracing Requested] property of the represented [Message].

```
[propput] HRESULT Trace(
    [in] long lTrace
);
```

lTrace: A long integer that specifies the trace method for the message. This parameter corresponds to the [MQMSGTRACE \(section 2.2.2.13\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the lTrace input parameter is NOT a valid **MQMSGTRACE** enumeration value:
 - The server MUST return MQ_ERROR_ILLEGAL_PROPERTY_VALUE (0x 0xC00E0018L), and take no further action.
- The [Tracing Requested] property of the represented [Message] MUST be set to the value contained in the lTrace input.

3.17.4.1.11 Priority (Opnum 17)

The **Priority** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Priority] property of the represented [Message].


```
[propget] HRESULT Priority(
    [out, retval] long* plPriority
);
```

plPriority: A pointer to a long integer that identifies priority of the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plPriority output parameter MUST be set to the value of the [Priority] property of the represented [Message].

3.17.4.1.12 Priority (Opnum 18)

The **Priority** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Priority] property of the represented [Message].

```
[propput] HRESULT Priority(
    [in] long lPriority
);
```

lPriority: A long integer that specifies the priority of the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the lPriority input parameter is NOT in the range $0 \leq \text{lPriority} \leq 7$:
 - The server MUST return MQ_ERROR_ILLEGAL_PROPERTY_VALUE (0x 0xC00E0018L), and take no further action.
- The [Priority] property of the represented [Message] MUST be set to the value contained in the lPriority input parameter.

3.17.4.1.13 Journal (Opnum 19)

The **Journal** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Journaling Requested] property of the represented [Message].

```
[propget] HRESULT Journal(
    [out, retval] long* plJournal
);
```

plJournal: A pointer to a long integer that identifies the journaling setting for the message. This parameter corresponds to the [MQMSGJOURNAL \(section 2.2.2.12\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The `pIJournal` output parameter MUST be set to the value of the [Journaling Requested] property of the represented [Message].

3.17.4.1.14 Journal (Opnum 20)

The **Journal** method is received by the server in an `RPC_REQUEST` packet. In response, the server sets the [Journaling Requested] property of the represented [Message].

```
[propput] HRESULT Journal(  
    [in] long lJournal  
);
```

IJournal: A long integer that specifies the journaling setting for the message. This parameter corresponds to the [MQMSGJOURNAL \(section 2.2.2.12\)](#) enum.

Return Values: The method MUST return `S_OK` (0x00000000) on success, or an implementation-specific error `HRESULT` on failure.

When processing this call, the server MUST follow these guidelines:

- If the `IJournal` input parameter is NOT a valid **MQMSGJOURNAL** enumeration value:
 - The server MUST return `MQ_ERROR_ILLEGAL_PROPERTY_VALUE` (0x 0xC00E0018L), and take no further action.
- The [Journaling Requested] property of the represented [Message] MUST be set to the value contained in the `IJournal` input parameter.

3.17.4.1.15 ResponseQueueInfo_v1 (Opnum 21)

The **ResponseQueueInfo_v1** method is received by the server in an `RPC_REQUEST` packet. In response, the server returns an [IMSMQQueueInfo](#) interface pointer to an [MSMQQueueInfo](#) object that represents the [Queue] identified by the [Response Queue Format Name] property of the represented [Message].

```
HRESULT ResponseQueueInfo_v1(  
    [out, retval] IMSMQQueueInfo** ppqinfoResponse  
);
```

ppqinfoResponse: A pointer to a pointer to an **IMSMQQueueInfo** interface, that upon successful completion will contain an [MSMQQueue](#) object representing the queue that will receive the response messages.

Return Values: The method MUST return `S_OK` (0x00000000) on success, or an implementation-specific error `HRESULT` on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new `MSMQQueueInfo` object instance.
- Set the `QueueFormatName` instance variable of the created `MSMQQueueInfo` object to the value of the [Response Queue Format Name] property of the represented [Message].

- Invoke `MSMQQueueInfo::Refresh` on the created `MSMQQueueInfo` instance.
- The `pqinfoResponse` output parameter MUST be set to a pointer to an **IMSMQQueueInfo** interface for the newly created `MSMQQueueInfo` object.

3.17.4.1.16 ResponseQueueInfo_v1 (Opnum 22)

The **ResponseQueueInfo_v1** method is received by the server in an `RPC_REQUEST` packet. In response, the server sets the [Response Queue Format Name] property of the represented [Message].

```
[putref] HRESULT ResponseQueueInfo_v1(
    [in] IMSMQQueueInfo* pqinfoResponse
);
```

pqinfoResponse: A pointer to an [IMSMQQueueInfo](#) interface for the [MSMQQueueInfo](#) object that represents the [Queue] that the receiving application may send response messages to.

Return Values: The method MUST return `S_OK` (0x00000000) on success, or an implementation-specific error `HRESULT` on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the `QueueFormatName` from the `pqinfoResponse` instance state.
- The [Response Queue Format Name] property of the represented [Message] MUST be set to the obtained format name.

3.17.4.1.17 AppSpecific (Opnum 23)

The **AppSpecific** method is received by the server in an `RPC_REQUEST` packet. In response, the server returns the application-specific [AppSpecific] property of the represented [Message].

```
[propget] HRESULT AppSpecific(
    [out, retval] long* plAppSpecific
);
```

plAppSpecific: A pointer to a long integer that contains the application-specific value for this message.

Return Values: The method MUST return `S_OK` (0x00000000) on success, or an implementation-specific error `HRESULT` on failure.

When processing this call, the server MUST follow these guidelines:

- The `plAppSpecific` output parameter MUST be set to the value of the [AppSpecific] property of the represented [Message].

3.17.4.1.18 AppSpecific (Opnum 24)

The **AppSpecific** method is received by the server in an `RPC_REQUEST` packet. In response, the server sets the application-specific [AppSpecific] property of the represented [Message].

```
[propput] HRESULT AppSpecific(
    [in] long lAppSpecific
);
```

lAppSpecific: A long integer that contains the application-specific value for the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [AppSpecific] property of the represented [Message] MUST be set to the value contained in the lAppSpecific input parameter.

3.17.4.1.19 SourceMachineGuid (Opnum 25)

The **SourceMachineGuid** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Source Machine GUID] property of the represented [Message].

```
[propget] HRESULT SourceMachineGuid(
    [out, retval] BSTR* pbstrGuidSrcMachine
);
```

pbstrGuidSrcMachine: A pointer to a BSTR that contains the GUID identifier of the computer that sent the message. The string MUST adhere to the following format, specified using Augmented BNF:

```
guid          = dword-part "-" word-part "-" word-part
               "-" 2byte-part "-" 6byte-part
dword-part    = 2word-part
word-part     = 2byte-part
byte-part     = 2hex-digit
hex-digit     = %x30-39 / %x41-46 / %x61-66
```

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The pbstrGuidSrcMachine output parameter MUST be set to the value of the [Source Machine GUID] property of the represented [Message].

3.17.4.1.20 BodyLength (Opnum 26)

The **BodyLength** method is received by the server in an RPC_REQUEST packet. In response, the server returns the number of bytes in the [Body] property of the represented [Message].

```
[propget] HRESULT BodyLength(
    [out, retval] long* pcbBody
);
```

pcbBody: A pointer to a long integer that contains the number of bytes in the body of the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The pcbBody output parameter MUST be set to the byte length of the [Body] property of the represented [Message].

3.17.4.1.21 Body (Opnum 27)

The **Body** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Body] property of the represented [Message].

```
[propget] HRESULT Body(  
    [out, retval] VARIANT* pvarBody  
);
```

pvarBody: A pointer to a VARIANT that contains the body content of the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The pvarBody output parameter MUST be set to the value of the [Body] property of the represented [Message].

3.17.4.1.22 Body (Opnum 28)

The **Body** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Body] property of the represented [Message].

```
[propput] HRESULT Body(  
    [in] VARIANT varBody  
);
```

varBody: A VARIANT that contains the body content of the message. The supported VARIANT types are:

```
VT_I1  
VT_UI1  
VT_I2  
VT_UI2  
VT_I4  
VT_UI4  
VT_R4  
VT_R8  
VT_CY  
VT_DATE  
VT_BOOL  
VT_BSTR
```

VT_UNKNOWN
VT_DISPATCH
VT_ARRAY | * (VT_ARRAY can be combined with any of the above types)

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the varBody input parameter is NOT a valid VARIANT type as described above:
 - The server MUST return E_INVALIDARG (0x80000003), and take no further action.
- The [Body] property of the represented [Message] MUST be set to the value contained in the varBody input parameter.

3.17.4.1.23 AdminQueueInfo_v1 (Opnum 29)

The **AdminQueueInfo_v1** method is received by the server in an RPC_REQUEST packet. In response, the server returns an **IMSMQQueueInfo** interface pointer to an **MSMQQueueInfo** object that represents the [Queue] that is identified by the [Administration Queue Format Name] property of the represented [Message].

```
HRESULT AdminQueueInfo_v1(  
    [out, retval] IMSMQQueueInfo** ppqinfoAdmin  
);
```

ppqinfoAdmin: A pointer to a to an **IMSMQQueueInfo** interface pointer, that upon successful completion will contain an **MSMQQueue** object representing the queue that will receive the administration messages sent by the system.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQQueueInfo object instance.
- Set the *QueueFormatName* instance variable of the created MSMQQueueInfo object to the value of the [Administration Queue Format Name] property of the represented [Message].
- Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.
- The ppqinfoAdmin output parameter MUST be set to a pointer to an **IMSMQQueueInfo** interface for the newly created MSMQQueueInfo object.

3.17.4.1.24 AdminQueueInfo (Opnum 30)

The **AdminQueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Administration Queue Format Name] property of the represented [Message].

```
[propputref] HRESULT AdminQueueInfo(  
    [in] IMSMQQueueInfo* pqinfoAdmin
```

);

pqinfoAdmin: A pointer to an [IMSMQQueueInfo](#) interface for the [MSMQQueueInfo](#) object that represents the **administration queue**.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the *QueueFormatName* from the pqinfoResponse instance state.
- The [Response Queue Format Name] property of the represented [Message] MUST be set to the obtained format name.

3.17.4.1.25 Id (Opnum 31)

The **Id** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Message Identifier] property of the represented [Message].

```
[propget] HRESULT Id(  
    [out, retval] VARIANT* pvarMsgId  
);
```

pvarMsgId: A pointer to a VARIANT that contains a 20 byte array (VT_ARRAY | VT_UI1) that contains the unique message identifier.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The pvarMsgId output parameter MUST be set to the value of the [Message Identifier] property of the represented [Message].

3.17.4.1.26 CorrelationId (Opnum 32)

The **CorrelationId** method is received by the server in an RPC_REQUEST packet. In response, the server sets the application-specific [Correlation Identifier] property of the represented [Message].

```
[propget] HRESULT CorrelationId(  
    [out, retval] VARIANT* pvarMsgId  
);
```

pvarMsgId: A VARIANT array of BYTES (VT_ARRAY | VT_UI1) that contains the application-specific [Correlation Identifier] for the [Message].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Correlation Identifier] property of the represented [Message] MUST be set to the value contained in the varMsgId input parameter.

3.17.4.1.27 CorrelationId (Opnum 33)

The **CorrelationId** method is received by the server in an RPC_REQUEST packet. In response, the server sets the application-specific [Correlation Identifier] property of the represented [Message].

```
[propput] HRESULT CorrelationId(
    [in] VARIANT varMsgId
);
```

varMsgId: A VARIANT array of BYTES (VT_ARRAY | VT_UI1) that contains the application-specific [Correlation Identifier] for the [Message].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Correlation Identifier] property of the represented [Message] MUST be set to the value contained in the varMsgId input parameter.

3.17.4.1.28 Ack (Opnum 34)

The **Ack** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Acknowledgements Requested] property of the represented [Message].

```
[propget] HRESULT Ack(
    [out, retval] long* plAck
);
```

plAck: A pointer to a long integer that identifies the type of acknowledgments requested for this message. This parameter corresponds to the [MQMSGACKNOWLEDGEMENT \(section 2.2.2.11\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plAck output parameter MUST be set to the value of the [Acknowledgements Requested] property of the represented [Message].

3.17.4.1.29 Ack (Opnum 35)

The **Ack** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Acknowledgments Requested] property of the represented [Message].

```
[propput] HRESULT Ack(
    [in] long lAck
);
```


IAck: A long integer that specifies the type of acknowledgments requested for the message.
This parameter corresponds to the [MQMSGACKNOWLEDGEMENT \(section 2.2.2.11\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the IAck input parameter is NOT a valid **MQMSGACKNOWLEDGEMENT** enumeration value:
 - The server MUST return MQ_ERROR_ILLEGAL_PROPERTY_VALUE (0x 0xC00E0018L) and take no further action.
- The [Acknowledgments Requested] property of the represented [Message] MUST be set to the value contained in the IAck input parameter.

3.17.4.1.30 Label (Opnum 36)

The **Label** method is received by the server in an RPC_REQUEST packet. In response, the server returns the application-specific [Label] property of the represented [Message].

```
[propget] HRESULT Label(  
    [out, retval] BSTR* pbstrLabel  
);
```

pbstrLabel: A pointer to a BSTR that contains the application defined string description of the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the [Label] property of the represented [Message] equals NULL
 - The server MUST set the pbstrLabel output parameter to an empty BSTR ("").
- Else
 - The pbstrLabel output parameter MUST contain the value of the [Label] property of the represented [Message].

3.17.4.1.31 Label (Opnum 37)

The **Label** method is received by the server in an RPC_REQUEST packet. In response, the server sets the application-specific [Label] property of the represented [Message].

```
[propput] HRESULT Label(  
    [in] BSTR bstrLabel  
);
```

bstrLabel: A BSTR that contains the application defined string description of the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- If the bstrLabel input parameter ≥ 250 characters
 - The server MUST return MQ_ERROR_LABEL_TOO_LONG (0xC00E005D) and take no further action.
- The [Label] property of the represented [Message] MUST be set to the value contained in the bstrLabel input parameter.

3.17.4.1.32 MaxTimeToReachQueue (Opnum 38)

The **MaxTimeToReachQueue** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Time To Be Received] property of the represented [Message].

```
[propget] HRESULT MaxTimeToReachQueue(  
    [out, retval] long* plMaxTimeToReachQueue  
);
```

plMaxTimeToReachQueue: A pointer to a long integer that contains the number of seconds allowed for the message to reach the destination queue; otherwise it is discarded.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plMaxTimeToReachQueue output parameter MUST be set to the total number of seconds for the [Time To Reach Queue] property of the represented [Message].

3.17.4.1.33 MaxTimeToReachQueue (Opnum 39)

The **MaxTimeToReachQueue** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Time To Reach Queue] property of the represented [Message].

```
[propput] HRESULT MaxTimeToReachQueue(  
    [in] long lMaxTimeToReachQueue  
);
```

lMaxTimeToReachQueue: A long integer that contains the number of seconds allowed for the message to reach the destination queue; otherwise it is discarded.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Time To Reach Queue] property of the represented [Message] MUST be set to the time duration value of the number of seconds specified in the lMaxTimeToReachQueue input parameter.

3.17.4.1.34 MaxTimeToReceive (Opnum 40)

The **MaxTimeToReceive** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Time To Be Received] property of the represented [Message].

```
[propget] HRESULT MaxTimeToReceive(  
    [out, retval] long* plMaxTimeToReceive  
);
```

plMaxTimeToReceive: A pointer to a long integer that contains the number of seconds allowed for the message to be received from the destination queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plMaxTimeToReceive output parameter MUST be set to the total number of seconds for the [Time To Be Received] property of the represented [Message].

3.17.4.1.35 MaxTimeToReceive (Opnum 41)

The **MaxTimeToReceive** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Time To Be Received] property of the represented [Message].

```
[propput] HRESULT MaxTimeToReceive(  
    [in] long lMaxTimeToReceive  
);
```

lMaxTimeToReceive: A long integer that contains the number of seconds allowed for the message to be received from the destination queue; otherwise it is discarded.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Time To Be Received] property of the represented [Message] MUST be set to the time duration value of the number of seconds specified in the lMaxTimeToReachQueue input parameter.

3.17.4.1.36 HashAlgorithm (Opnum 42)

The **HashAlgorithm** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Hash Algorithm] property of the represented [Message].

```
[propget] HRESULT HashAlgorithm(  
    [out, retval] long* plHashAlg  
);
```

plHashAlg: A pointer to a long integer that identifies the type of hash algorithm that will be used. This parameter corresponds to the [MQCALG \(section 2.2.2.18\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plHashAlg output parameter MUST be set to the value of the [Hash Algorithm] property of the represented [Message].

3.17.4.1.37 HashAlgorithm (Opnum 43)

The **HashAlgorithm** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Hash Algorithm] property of the represented [Message].

```
[propput] HRESULT HashAlgorithm(  
    [out, retval] long* plHashAlg  
);
```

plHashAlg: A long integer that identifies the type of hash algorithm that will be used. This parameter corresponds to the [MQCALG \(section 2.2.2.18\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Hash Algorithm] property of the represented [Message] MUST be set to the value contained in the lHashAlg input parameter.

3.17.4.1.38 EncryptAlgorithm (Opnum 44)

The **EncryptAlgorithm** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Encryption Algorithm] property of the represented [Message].

```
[propget] HRESULT EncryptAlgorithm(  
    [out, retval] long* plEncryptAlg  
);
```

plEncryptAlg: A pointer to a long integer that indicates the algorithm employed by the message transfer process to encrypt or decrypt the [Body]. This value is not interpreted by this protocol; rather, the message transfer process interprets and validates the value. Potential values are contained in the [MQCALG \(section 2.2.2.18\)](#) enumeration which assigns numeric values to specific cryptographic algorithms. The message transfer implementation MAY support a subset of the encryption algorithms defined by **MQCALG** (section 2.2.2.18).<154>

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plEncryptAlg output parameter MUST be set to the value of the [Encryption Algorithm] property of the represented [Message].

3.17.4.1.39 EncryptAlgorithm (Opnum 45)

The **EncryptAlgorithm** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Encryption Algorithm] property of the represented [Message].

```
[propput] HRESULT EncryptAlgorithm(  
    [in] long lEncryptAlg  
);
```

IEncryptAlg: A long integer that indicates the algorithm employed by the message transfer process to encrypt or decrypt the [Body]. This value is not interpreted by this protocol; rather, the message transfer process interprets and validates the value. Potential values are contained in the [MQCALG \(section 2.2.2.18\)](#) enumeration which assigns numeric values to specific cryptographic algorithms. The message transfer implementation MAY support a subset of the encryption algorithms defined by **MQCALG** (section 2.2.2.18).[<155>](#)

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Encryption Algorithm] property of the represented [Message] MUST be set to the value contained in the lEncryptAlg input parameter.

3.17.4.1.40 SentTime (Opnum 46)

The **SentTime** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Sent Time] property of the represented [Message].

```
[propget] HRESULT SentTime(  
    [out, retval] VARIANT* pvarSentTime  
);
```

pvarSentTime: A pointer to a VARIANT that contains a VT_DATE that contains the UTC date and timestamp of when the message was sent.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The pvarSentTime output parameter MUST be set to the value of the [Sent Time] property of the represented [Message].

3.17.4.1.41 ArrivedTime (Opnum 47)

The **ArrivedTime** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Arrived Time] property of the represented [Message].

```
[propget] HRESULT ArrivedTime(  
    [out, retval] VARIANT* plArrivedTime  
);
```

plArrivedTime: A pointer to a VARIANT that contains a VT_DATE that contains the UTC date and timestamp of when the message arrived in the queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The plArrivedTime output parameter MUST be set to the value of the [Arrived Time] property of the represented [Message].

3.17.4.1.42 DestinationQueueInfo (Opnum 48)

The **DestinationQueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server returns an [IMSMQQueueInfo](#) interface pointer to an [MSMQQueueInfo](#) object that represents the [Queue] identified by the [Destination Queue Format Name] property of the represented [Message].

```
[propget] HRESULT DestinationQueueInfo(  
    [out, retval] IMSMQQueueInfo4** ppqinfoDest  
);
```

ppqinfoDest: A pointer to a pointer to an **IMSMQQueueInfo** interface that upon successful completion will contain an [MSMQQueue](#) object that represents the destination queue for the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQQueueInfo object instance.
- Set the *QueueFormatName* instance variable of the created MSMQQueueInfo object to the value of the [Destination Queue Format Name] property of the represented [Message].
- Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.
- The pqinfoAdmin output parameter MUST be set to a pointer to an **IMSMQQueueInfo** interface for the newly created MSMQQueueInfo object.

3.17.4.1.43 SenderCertificate (Opnum 49)

The **SenderCertificate** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Sender Certificate] property of the represented [Message].

```
[propget] HRESULT SenderCertificate(  
    [out, retval] VARIANT* pvarSenderCert  
);
```

pvarSenderCert: A pointer to a VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the user certificate.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The pvarSenderCert output parameter MUST be set to the value of the [Sender Certificate] property of the represented [Message].

3.17.4.1.44 SenderCertificate (Opnum 50)

The **SenderCertificate** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Sender Certificate] property of the represented [Message].

```
[propput] HRESULT SenderCertificate(  
    [in] VARIANT varSenderCert  
);
```

varSenderCert: A VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the user certificate. The certificate MUST be an X.509-encoded certificate, as specified in [\[RFC3280\]](#).

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Sender Certificate] property of the represented [Message] MUST be set to the value contained in the varSenderCert input parameter.

3.17.4.1.45 SenderId (Opnum 51)

The **SenderId** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Sender Identifier] property of the represented [Message].

```
[propget] HRESULT SenderId(  
    [out, retval] VARIANT* pvarSenderId  
);
```

pvarSenderId: A pointer to a VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the sending user's identifier.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The pvarSenderId output parameter MUST be set to the value of the [Sender Identifier] property of the represented [Message].

3.17.4.1.46 SenderIdType (Opnum 52)

The **SenderIdType** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Sender Identifier Type] property of the represented [Message].

```
[propget] HRESULT SenderIdType(
    [out, retval] long* plSenderIdType
);
```

plSenderIdType: A pointer to a LONG that identifies the type of identifier that is stored with the message. This parameter corresponds to the [MQMSGSENDERIDTYPE \(section 2.2.2.14\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *plSenderIdType* output parameter MUST be set to the value of the [Sender Identifier Type] property of the represented [Message].

3.17.4.1.47 SenderIdType (Opnum 53)

The **SenderIdType** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Sender Identifier Type] property of the represented [Message].

```
[propput] HRESULT SenderIdType(
    [in] long lSenderIdType
);
```

lSenderIdType: A long integer that identifies the type of identifier that is stored with the message. This parameter corresponds to the [MQMSGSENDERIDTYPE](#) enum as defined in section [2.2.2.14](#).

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation specific error HRESULT on failure.

When processing this call the server MUST follow these guidelines:

- The [Sender Identifier] property of the represented [Message] MUST be set to the value contained in the lSenderIdType input parameter.

3.17.4.1.48 Send (Opnum 54)

The **Send** method is received by the server in an RPC_REQUEST packet. In response, the server sends a message.

```
HRESULT Send(
    [in] IDispatch* DestinationQueue,
    [in, optional] VARIANT* Transaction
);
```

DestinationQueue: A pointer to an IDispatch interface that can reference either an MSMQQueue object instance or an MSMQDestination object instance.

Transaction: A pointer to a VARIANT that contains either:

- A VT_DISPATCH or a VT_DISPATCH | VT_BYREF that points to an MSMQTransaction object

- A VT_I4 that references one of the following enumeration values as defined in section [2.2.2.1](#).

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *DestinationQueue* input parameter is NULL:
 - The server MUST return E_INVALIDARG (0x80000003) and take no further action.
- If either [Class] OR [Sender Identifier] OR [Symmetric Key] OR [Signature] OR [Authentication Provider Name] properties of the represented [Message] were set after this object was initialized AND the [Connector Type GUID] property of the represented [Message] equals the NULL GUID ({00000000-0000-0000-0000-000000000000}):
 - The server MUST return MQ_ERROR_MISSING_CONNECTOR_TYPE (0xC00E0055) and take no further action.
- If ([Authentication Provider Type] of the represented [Message] is not NULL AND [Authentication Provider Name] of the represented [Message] is NULL) OR ([Authentication Provider Type] of the represented [Message] is NULL AND [Authentication Provider Name] of the represented [Message] is not NULL):
 - The server MUST return MQ_ERROR_INSUFFICIENT_PROPERTIES (0xC00E003F) and take no further action.
- Define *transactional send* as a Boolean value set to False.
- If the *Transaction* input parameter is NOT NULL:
 - If the *Transaction* input parameter is VT_DISPATCH or VT_DISPATCH | VT_BYREF:
 - Define *transaction identifier* as the value of the *TransactionIdentifier* instance variable of the ITransaction object obtained by MSMQTransaction::get_ITransaction method on the MSMQTransaction object obtained from the *Transaction* input parameter.
 - Set *transactional send* to True.
 - Else, if the *Transaction* input parameter is VT_I4:
 - If the numeric value represented by the *Transaction* input parameter equals either MQ_MTS_TRANSACTION (0x00000001) or MQ_XA_TRANSACTION (0x00000002).
 - Define and retrieve *transaction identifier* as described in section [2.2.2.1](#), using the enum numeric value represented by the *Transaction* input parameter.
 - Set *transactional send* to True.
 - Else:
 - The server MUST return an error HRESULT and take no further action.
- If *transactional send* is True:

- Identify the [Enlisted Transaction] from the [Enlisted Transactions Table] of the [Local Queue Manager] where the value of [Transactional Message Sequence Identifier] property of the [Enlisted Transaction] equals the value of the *transaction identifier*.
- If an [Enlisted Transaction] cannot be located:
 - Create new [Enlisted Transaction] and set the [Transactional Message Sequence Identifier] property to the value of the *transaction identifier*. Refer to this [Enlisted Transaction] as the identified [Enlisted Transaction] henceforth.
 - Add the created [Enlisted Transaction] to the [Enlisted Transactions Table] of the [Local Queue Manager].
- Create a new [Transactional Operation] and set:
 - The [Message Reference] property with the represented [Message].
 - The [Operation Type] property to the [Operation Type].Send enumeration value.
- Add the newly-created [Transaction Operation] to the [Transactional Operations List] of the [Enlisted Transaction] identified above.
- Set the [IsLocked] property of the represented [Message] to True.
- Set the [Transactional Message Sequence Identifier] property of the represented [Message] to the value of the *transaction identifier*.
- If the *SenderIdCache* instance variable equals NULL
 - If the [Sender Identifier] property of the represented [Message] equals NULL
 - Set the [Sender Identifier] property of the represented [Message] to the SID of the caller. [<156>](#)
- Else
 - Set the [Sender Identifier] property of the represented [Message] to the value of the *SenderIdCache* instance variable.
- If the *SenderCertificateCache* instance variable equals NULL
 - If the [Sender Certificate] property of the represented [Message] equals NULL:
 - Set the [Sender Identifier] property of the represented [Message] to the **internal certificate** from the internal certificate store for the user identified by the [Sender Identifier] property of the represented [Message].
 - Else
 - Look up the certificate in the personal certificate store using the [Sender Certificate] property value.
 - If not found
 - The server MUST return an error and take no further action.
- Else

- Set the [Sender Certificate] property of the represented [Message] to the value of the *SenderCertificateCache*.
- If the *SymmetricKeyCache* instance variable is NOT NULL:
 - Set the [Symmetric Key] property of the represented [Message] to the value of the *SymmetricKeyCache* instance variable.
- Set the [Source Machine GUID] property of the represented [Message] to the [GUID] property of the [Local Queue Manager].
- If the [Connector Type GUID] property of the represented [Message] does NOT equal the NULL GUID ({00000000-0000-0000-0000-000000000000}):
 - Set the [Class] property value of the represented [Message] to MQMSG_CLASS_NORMAL (0x00000000).
- Set the [Message Identifier] property of the represented [Message] to a new and unique value as specified by [\[MS-MQMQ\]](#) section 2.2.8.
- Define *QueueFormatName* as follows:
 - If the *DestinationQueue* input parameter represents an MSMQQueue object:
 - Set *QueueFormatName* to the result of calling MSMQQueueInfo::get_FormatName on the MSMQQueueInfo object returned as a result of invoking MSMQQueue::get_QueueInfo method on the MSMQQueue object represented by the *DestinationQueue* input parameter.
 - If the *DestinationQueue* input parameter represents an MSMQDestination object:
 - Set *QueueFormatName* to the result of calling the MSMQDestination::get_FormatName method on the MSMQDestination object represented by the *DestinationQueue* input parameter.
- Set the [Destination Queue Format Name] property of the represented [Message] to *QueueFormatName*.
- Set the [Sent Time] property of the represented [Message] to the current time.
- If the [Time To Reach Queue] property of the represented [Message] is greater than [Time To Be Received]:
 - Set the [Time To Be Received] to the same value as [Time To Reach Queue].
- If the [Connector Type GUID] property of the represented [Message] equals the NULL GUID ({00000000-0000-0000-0000-000000000000}):
 - If the *RequestedAuthenticationLevel* instance variable is NOT equal to MQMSG_AUTH_LEVEL_NONE (0x00000000)
 - Calculate the hash value using the specified [Hash Algorithm] property of the represented [Message] and in the manner identified by the *RequestedAuthenticationLevel* instance variable.
 - Obtain the private key from the internal certificate store that is associated with the [Sender Identifier] value.

- Set the [Signature] property of the represented [Message] to the signature calculated using the specified [Authentication Provider Type] and [Sender Certificate] properties of the represented [Message] and the obtained private key applied to the calculated hash value.
- Set the [Authentication Level] property of the represented [Message] to the associated value from the *RequestedAuthenticationLevel* instance variable. Refer to section [2.2.2.16](#) for details of the translation.
- If the *SOAPBody* instance variable is NOT NULL:
 - The server MUST append the *SOAPBody* instance variable value to the end of the [SOAP Body] property of the represented [Message].
- If the *SOAPHeader* instance variable is NOT NULL:
 - The server MUST append the *SOAPHeader* instance variable value to the end of the [SOAP Header] property of the represented [Message].
- Define *TargetOpenQueue* as follows:
 - If the *DestinationQueue* input parameter represents an MSMQQueue object:
 - Set *TargetOpenQueue* to the *OpenQueueReference* instance variable of the MSMQQueue object represented by the *DestinationQueue* input parameter.
 - If the *DestinationQueue* input parameter represents an MSMQDestination object:
 - Set *TargetOpenQueue* to the [Open Queue] in the [Open Queues Table] of the [Local Queue Manager] identified by the *QueueHandle* instance variable of the MSMQDestination object represented by the *DestinationQueue* input parameter.
- Insert the represented [Message] into the [Messages Table] of the [Queue] identified by the [Queue Reference] property of the *TargetOpenQueue*.

3.17.4.1.49 AttachCurrentSecurityContext (Opnum 55)

The **AttachCurrentSecurityContext** method is received by the server in an RPC_REQUEST packet. In response, the server caches the relevant information required to sign a message on behalf of the client; including the [Sender Identifier], [Sender Certificate] and [Symmetric Key]. This method is provided purely as an optimization to allow the client to reduce lookups of the security information about the calling client each time the message is sent. If any of these properties are not specified by the client, the server performs the necessary lookups and caches the results when this method is called. The [Sender Identifier], [Sender Certificate] and [Symmetric Key] property values of the represented [Message] MUST NOT be updated as a result of calling this method. This method is superseded by *IMSMQMessage4::AttachSecurityContext2*.

```
HRESULT AttachCurrentSecurityContext();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the [Sender Identifier] property of the represented [Message] equals NULL:

- Set the *SenderIdCache* instance variable to the SID of the caller. <157>
- Else:
 - Set the *SenderIdCache* instance variable to the value of the [Sender Identifier] property of the represented [Message].
- If the [Sender Certificate] property of the represented [Message] equals NULL:
 - Set the [Sender Identifier] property of the represented [Message] to the [Internal Certificate] from the internal certificate store for the user identified by the [Sender Identifier] property of the represented [Message].
- Else
 - Look up the certificate in the personal certificate store using the [Sender Certificate] property value.
 - If found
 - Set the *SenderCertificateCache* instance variable to the value of the [Sender Certificate] property of the represented [Message].
 - Else
 - The server MUST return an error and take no further action.
- If the [Symmetric Key] property of the represented [Message] is NOT NULL:
 - Set the *SymmetricKeyCache* instance variable to the value of the [Symmetric Key] property of the represented [Message].

3.17.4.1.50 SenderVersion (Opnum 56)

The **SenderVersion** method is received by the server in an RPC_REQUEST packet. In response, the server returns a numeric value that indicates the version of transfer used to send the message.

```
[propget] HRESULT SenderVersion(
    [out, retval] long* plSenderVersion
);
```

plSenderVersion: A pointer to a long integer that specifies the version of transfer used to send the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *plSenderVersion* output parameter MUST be set to (0x00000010).

3.17.4.1.51 Extension (Opnum 57)

The **Extension** method is received by the server in an RPC_REQUEST packet. In response, the server returns the application-specific [Extension] property of the represented [Message].

```
[propget] HRESULT Extension(
    [out, retval] VARIANT* pvarExtension
);
```

pvarExtension: A pointer to a VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the application-specific information.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pvarExtension* output parameter MUST be set to the value of the [Extension] property of the represented [Message].

3.17.4.1.52 Extension (Opnum 58)

The **Extension** method is received by the server in an RPC_REQUEST packet. In response, the server sets the application-specific [Extension] property of the represented [Message].

```
[propput] HRESULT Extension(
    [in] VARIANT varExtension
);
```

varExtension: A VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains application-specific information.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Extension] property of the represented [Message] MUST be set to the value contained in the *varExtension* input parameter.

3.17.4.1.53 ConnectorTypeGuid (Opnum 59)

The **ConnectorTypeGuid** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Connector Type GUID] property of the represented [Message].

```
[propget] HRESULT ConnectorTypeGuid(
    [out, retval] BSTR* pbstrGuidConnectorType
);
```

pbstrGuidConnectorType: A pointer to a BSTR that contains the GUID stored with the message. The string MUST adhere to the following format, specified using Augmented BNF:

```
guid          = "{" dword-part "-" word-part "-" word-part
                "-" 2byte-part "-" 6byte-part "}"

dword-part    = 2word-part
word-part     = 2byte-part
```

```

byte-part  = 2hex-digit
hex-digit  = %x30-39 / %x41-46 / %x61-66

```

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pbstrGuidConnectorType* output parameter MUST be set to the value of the [Connector Type GUID] property of the represented [Message].

3.17.4.1.54 ConnectorTypeGuid (Opnum 60)

The **ConnectorTypeGuid** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Connector Type GUID] property of the represented [Message].

```

[propput] HRESULT ConnectorTypeGuid(
    [in] BSTR bstrGuidConnectorType
);

```

bstrGuidConnectorType: A pointer to a BSTR that contains the GUID that will be stored with the message. The string MUST adhere to the following format, specified using Augmented BNF:

```

guid          = "{" dword-part "-" word-part "-" word-part
                "-" 2byte-part "-" 6byte-part "}"

dword-part    = 2word-part
word-part     = 2byte-part
byte-part     = 2hex-digit
hex-digit     = %x30-39 / %x41-46 / %x61-66

```

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the *bstrGuidConnectorType* input parameter is NOT compliant with the format above
 - The server MUST return MQ_ERROR_ILLEGAL_PROPERTY_VALUE (0x 0xC00E0018L) and take no further action.
- The [Connector Type GUID] property of the represented [Message] MUST be set to the value contained in the *bstrGuidConnectorType* input parameter.

3.17.4.1.55 TransactionStatusQueueInfo (Opnum 61)

The **TransactionStatusQueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server returns an IMSMQQueueInfo interface pointer to an MSMQQueueInfo object that represents the [Queue] identified by the [Transaction Status Format Name] property of the represented [Message].

```
[propget] HRESULT TransactionStatusQueueInfo(
    [out, retval] IMSMQQueueInfo4** ppqinfoXactStatus
);
```

ppqinfoXactStatus: A pointer to an IMSMQQueueInfo interface pointer that upon successful completion will contain an MSMQQueue object representing the **transaction status queue** for the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQQueueInfo object instance.
- Set the *QueueFormatName* instance variable of the created MSMQQueueInfo object to the value of the [Transaction Status Format Name] property of the represented [Message].
- Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.
- The *ppqinfoXactStatus* output parameter MUST be set to a pointer to an IMSMQQueueInfo interface for the newly created MSMQQueueInfo object.

3.17.4.1.56 DestinationSymmetricKey (Opnum 62)

The **DestinationSymmetricKey** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Symmetric Key] property of the represented [Message].

```
[propget] HRESULT DestinationSymmetricKey(
    [out, retval] VARIANT* pvarDestSymmKey
);
```

pvarDestSymmKey: A pointer to a VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the symmetric key.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pvarDestSymmKey* output parameter MUST be set to the value of the [Symmetric Key] property of the represented [Message].

3.17.4.1.57 DestinationSymmetricKey (Opnum 63)

The **DestinationSymmetricKey** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Symmetric Key] property of the represented [Message]

```
[propput] HRESULT DestinationSymmetricKey(
    [in] VARIANT varDestSymmKey
);
```


varDestSymmKey: A VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the symmetric key.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Symmetric Key] property of the represented [Message] MUST be set to the value contained in *varDestSymmKey* input parameter.

3.17.4.1.58 Signature (Opnum 64)

The **Signature** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Signature] property of the represented [Message].

```
[propget] HRESULT Signature(  
    [out, retval] VARIANT* pvarSignature  
);
```

pvarSignature: A pointer to a VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the digital signature.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pvarSignature* output parameter MUST be set to the value of the [Signature] property of the represented [Message].

3.17.4.1.59 Signature (Opnum 65)

The **Signature** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Signature] property of the represented [Message].

```
[propput] HRESULT Signature(  
    [in] VARIANT varSignature  
);
```

varSignature: A VARIANT that contains a byte array (VT_ARRAY | VT_UI1) that contains the digital signature.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Signature] property of the represented [Message] MUST be set to the value contained in the *varSignature* input parameter.

3.17.4.1.60 AuthenticationProviderType (Opnum 66)

The **AuthenticationProviderType** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Authentication Provider Type] property of the represented [Message].

```
[propget] HRESULT AuthenticationProviderType(  
    [out, retval] long* plAuthProvType  
);
```

plAuthProvType: A pointer to a long integer that identifies the type of cryptographic provider that is used to generate the digital signature.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *plAuthProvType* output parameter MUST be set to the value of the [Authentication Provider Type] property of the represented [Message].

3.17.4.1.61 AuthenticationProviderType (Opnum 67)

The **AuthenticationProviderType** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Authentication Provider Type] property of the represented [Message].

```
[propput] HRESULT AuthenticationProviderType(  
    [in] long lAuthProvType  
);
```

lAuthProvType: A long integer that identifies the type of cryptographic provider that is used to generate the digital signature.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Authentication Provider Type] property of the represented [Message] MUST be set to the value contained in the *lAuthProvType* input parameter.

3.17.4.1.62 AuthenticationProviderName (Opnum 68)

The **AuthenticationProviderName** method is received by the server in an RPC_REQUEST packet. In response, the server returns the name [Authentication Provider Name] property of the represented [Message].

```
[propget] HRESULT AuthenticationProviderName(  
    [out, retval] BSTR* pbstrAuthProvName  
);
```

pbstrAuthProvName: A pointer to a BSTR that contains the descriptive name of the cryptographic provider. The valid values are:

Cryptographic provider	Version validity
"Microsoft Base Cryptographic Provider, v1,0"	Valid for all versions of MSMQ
"Microsoft Enhanced Cryptographic Provider, v1,0"	NOT valid for MSMQ 1.0 or MSMQ 2.0

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the [Authentication Provider Name] equals NULL
 - The pbstrAuthProvName output parameter MUST be set to an empty BSTR ("").
- Else
 - The pbstrAuthProvName output parameter MUST be set to the value of the [Authentication Provider Name] property of the represented [Message].

3.17.4.1.63 AuthenticationProviderName (Opnum 69)

The **AuthenticationProviderName** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Authentication Provider Name] property of the represented [Message].

```
[propput] HRESULT AuthenticationProviderName(  
    [in] BSTR bstrAuthProvName  
);
```

bstrAuthProvName: A BSTR that contains the descriptive name of the cryptographic provider. The valid values are:

Cryptographic provider	Version validity
"Microsoft Base Cryptographic Provider, v1,0"	Valid for all versions of MSMQ
"Microsoft Enhanced Cryptographic Provider, v1,0"	NOT valid for MSMQ 1.0 or MSMQ 2.0

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Authentication Provider Name] property of the represented [Message] MUST be set to the value contained in the bstrAuthProvName input parameter.

3.17.4.1.64 SenderId (Opnum 70)

The **SenderId** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Sender Identifier] property of the represented [Message].

```
[propput] HRESULT SenderId(
    [out, retval] VARIANT* pvarSenderId
);
```

pvarSenderId: A VARIANT that contains a byte array (VT_ARRAY | VT_UI1) containing the sender identifier for the user that is sending the [Message].

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Sender Identifier] property of the represented [Message] MUST be set to the value of the *varSenderId* input parameter.

3.17.4.1.65 MsgClass (Opnum 71)

The **MsgClass** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Class] property of the represented [Message].

```
[propget] HRESULT MsgClass(
    [out, retval] long* plMsgClass
);
```

plMsgClass: A pointer to a LONG that identifies the message type. This parameter corresponds to the [MQMSGCLASS \(section 2.2.2.9\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *plMsgClass* output parameter MUST be set to the value of the [Class] property of the represented [Message].

3.17.4.1.66 MsgClass (Opnum 72)

The **MsgClass** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Class] property of the represented [Message].

```
[propput] HRESULT MsgClass(
    [in] long lMsgClass
);
```

lMsgClass: A long integer that identifies the message type. This parameter corresponds to the [MQMSGCLASS \(section 2.2.2.9\)](#) enum.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The [Class] property of the represented [Message] MUST contain the value of the *IMsgClass* input parameter.

3.17.4.1.67 Properties (Opnum 73)

The **Properties** method is not implemented.

```
[propget] HRESULT Properties(
    [out, retval] IDispatch** ppcolProperties
);
```

ppcolProperties: A pointer to an IDispatch pointer. The server MUST ignore this parameter.

Return Values: The server MUST return E_NOTIMPL (0x80004001);

The server MUST take no action and return E_NOTIMPL (0x80004001).

3.17.4.1.68 TransactionId (Opnum 74)

The **TransactionId** method is received by the server in an RPC_REQUEST packet. In response, the server returns [Transactional Message Sequence Identifier] property of the represented [Message].

```
[propget] HRESULT TransactionId(
    [out, retval] VARIANT* pvarXactId
);
```

pvarXactId: A pointer to a VARIANT that contains a 20 byte array (VT_ARRAY | VT_UI1) containing the transaction identifier. The format for this ID MUST follow the structure as defined in [\[MS_MOMQ\]](#) section 2.2.8 where the Lineage part is the GUID of the sending computer and the Uniquifier part is a transaction sequence number.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pvarXactId* output parameter MUST be set to the value of the [Transactional Message Sequence Identifier] property of the represented [Message].

3.17.4.1.69 IsFirstInTransaction (Opnum 75)

The **IsFirstInTransaction** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IsFirstInTransaction] property of the represented [Message].

```
[propget] HRESULT IsFirstInTransaction(
    [out, retval] short* pisFirstInXact
);
```

pisFirstInXact: A pointer to a short that specifies whether or not the message was the first sent by the transaction.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pisFirstInXact* output parameter MUST be set to the value of the [IsFirstInTransaction] property of the represented [Message].

3.17.4.1.70 IsLastInTransaction (Opnum 76)

The **IsLastInTransaction** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IsLastInTransaction] property of the represented [Message].

```
[propget] HRESULT IsLastInTransaction(  
    [out, retval] short* pisLastInXact  
);
```

pisLastInXact: A pointer to a short that specifies whether or not the message was the last sent by the transaction.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pisLastInXact* output parameter MUST be set to the value of the [IsLastInTransaction] property of the represented [Message].

3.17.4.1.71 ResponseQueueInfo_v2 (Opnum 77)

The **ResponseQueueInfo_v2** method is received by the server in an RPC_REQUEST packet. In response, the server returns an IMSMQQueueInfo2 interface pointer to an MSMQQueueInfo object that represents the [Queue] identified by the [Response Queue Format Name] property of the represented [Message].

```
HRESULT ResponseQueueInfo_v2(  
    [out, retval] IMSMQQueueInfo2** ppqinfoResponse  
);
```

ppqinfoResponse: A pointer to a pointer to an IMSMQQueueInfo2 interface that upon successful completion will contain an MSMQQueue object representing the queue that will receive the response messages.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQQueueInfo object instance.
- Set the *QueueFormatName* instance variable of the created MSMQQueueInfo object to the value of the [Response Queue Format Name] property of the represented [Message].
- Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.

- The *pqinfoResponse* output parameter MUST be set to a pointer to an IMSMQQueueInfo2 interface for the newly created MSMQQueueInfo object.

3.17.4.1.72 ResponseQueueInfo_v2 (Opnum 78)

The **ResponseQueueInfo_v2** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Response Queue Format Name] property of the represented [Message].

```
HRESULT ResponseQueueInfo_v2 (
    [in] IMSMQQueueInfo2* pqinfoResponse
);
```

pqinfoResponse: A pointer to an IMSMQQueueInfo2 interface for the MSMQQueueInfo object representing the response queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the *QueueFormatName* from the pqinfoResponse instance state.
- The [Response Queue Format Name] property of the represented [Message] MUST be set to the obtained format name.

3.17.4.1.73 AdminQueueInfo_v2 (Opnum 79)

The **AdminQueueInfo_v2** method is received by the server in an RPC_REQUEST packet. In response, the server returns an IMSMQQueueInfo2 interface pointer to an MSMQQueueInfo object that represents the [Queue] identified by the [Administration Queue Format Name] property of the represented [Message].

```
HRESULT AdminQueueInfo_v2 (
    [out, retval] IMSMQQueueInfo2** ppqinfoAdmin
);
```

ppqinfoAdmin: A pointer to a pointer to an IMSMQQueueInfo2 interface that upon successful completion will contain an MSMQQueue object representing the queue that will receive administrative acknowledgement messages generated by the system.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQQueueInfo object instance
- Set the *QueueFormatName* instance variable of the created MSMQQueueInfo object to the value of the [Administration Queue Format Name] property of the represented [Message].
- Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.

- The *ppqinfoAdmin* output parameter MUST be set to a pointer to an IMSMQQueueInfo2 interface for the newly created MSMQQueueInfo object.

3.17.4.1.74 AdminQueueInfo_v2 (Opnum 80)

The **AdminQueueInfo_v2** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Administration Queue Format Name] property of the represented [Message].

```
HRESULT AdminQueueInfo_v2(
    [in] IMSMQQueueInfo2* ppqinfoAdmin
);
```

ppqinfoAdmin: A pointer to an IMSMQQueueInfo2 interface for the MSMQQueueInfo object representing the administration queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the *QueueFormatName* from the *ppqinfoAdmin* instance state.
- The [Administration Queue Format Name] property of the represented [Message] MUST be set to the obtained format name.

3.17.4.1.75 ReceivedAuthenticationLevel (Opnum 81)

The **ReceivedAuthenticationLevel** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Authentication Level] property of the represented [Message].

```
[propget] HRESULT ReceivedAuthenticationLevel(
    [out, retval] short* psReceivedAuthenticationLevel
);
```

psReceivedAuthenticationLevel: A pointer to a SHORT that identifies if the message was authenticated and what digital signature was used. This parameter corresponds to the MQMSGAUTHENTICATION enum as defined in section [MQMSGAUTHENTICATION](#).

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *psReceivedAuthenticationLevel* output parameter MUST be set to the value of the [Authentication Level] property of the represented [Message].

3.17.4.1.76 ResponseQueueInfo (Opnum 82)

The **ResponseQueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server returns an IMSMQQueueInfo4 interface pointer to an MSMQQueueInfo object that represents the [Queue] identified by the [Response Queue Format Name] property of the represented [Message].


```
[propget] HRESULT ResponseQueueInfo(
    [out, retval] IMSMQQueueInfo4** ppqinfoResponse
);
```

ppqinfoResponse: A pointer to a pointer to an IMSMQQueueInfo4 interface that upon successful completion will contain an MSMQQueue object representing the queue that will receive the response messages.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQQueueInfo object instance.
- Set the *QueueFormatName* instance variable of the created MSMQQueueInfo object to the value of the [Response Queue Format Name] property of the represented [Message].
- Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.
- The *pqinfoResponse* output parameter MUST be set to a pointer to an IMSMQQueueInfo4 interface for the newly created MSMQQueueInfo object.

3.17.4.1.77 ResponseQueueInfo (Opnum 83)

The **ResponseQueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Response Queue Format Name] property of the represented [Message].

```
[propputref] HRESULT ResponseQueueInfo(
    [in] IMSMQQueueInfo* pqinfoResponse
);
```

pqinfoResponse: A pointer to an IMSMQQueueInfo4 interface for the MSMQQueueInfo object representing the Response Queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the *QueueFormatName* from the pqinfoResponse instance state
- The [Response Queue Format Name] property of the represented [Message] MUST be set to the obtained format name.

3.17.4.1.78 AdminQueueInfo (Opnum 84)

The **AdminQueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server returns an IMSMQQueueInfo4 interface pointer to an MSMQQueueInfo object that represents the [Queue] identified by the [Administration Queue Format Name] property of the represented [Message].

```
[propget] HRESULT AdminQueueInfo(
```

```
[out, retval] IMSMQQueueInfo4** ppqinfoAdmin
);
```

ppqinfoAdmin: A pointer to a pointer to an IMSMQQueueInfo4 interface that upon successful completion will contain an MSMQQueue object representing the queue that will receive administrative acknowledgement messages generated by the system.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQQueueInfo object instance
- Set the *QueueFormatName* instance variable of the created MSMQQueueInfo object to the value of the [Administration Queue Format Name] property of the represented [Message].
- Invoke MSMQQueueInfo::Refresh on the created MSMQQueueInfo instance.
- The *pqinfoAdmin* output parameter MUST be set to a pointer to an IMSMQQueueInfo4 interface for the newly created MSMQQueueInfo object.

3.17.4.1.79 AdminQueueInfo (Opnum 85)

The **AdminQueueInfo** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Administration Queue Format Name] property of the represented [Message].

```
[propputref] HRESULT AdminQueueInfo(
    [in] IMSMQQueueInfo4* pqinfoAdmin
);
```

pqinfoAdmin: A pointer to an IMSMQQueueInfo4 interface for the MSMQQueueInfo object representing the administration queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the *QueueFormatName* from the pqinfoAdmin instance state.
- The [Administration Queue Format Name] property of the represented [Message] MUST be set to obtained format name.

3.17.4.1.80 ResponseDestination (Opnum 86)

The **ResponseDestination** method is received by the server in an RPC_REQUEST packet. In response, the server sets the [Administration Queue Format Name] property of the represented [Message].

```
[propget] HRESULT ResponseDestination(
    [out, retval] IDispatch** ppdestResponse
);
```

ppdestResponse: A pointer to an IMSMQQueueInfo4 interface for the MSMQQueueInfo object representing the administration queue.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the *QueueFormatName* from the pqinfoAdmin instance state.
- The [Administration Queue Format Name] property of the represented [Message] MUST be set to obtained format name.

3.17.4.1.81 ResponseDestination (Opnum 87)

The **ResponseDestination** method is received by the server in an RPC_REQUEST packet. In response, the server Sets the [Response Queue Format Name] property of the represented [Message].

```
[propputref] HRESULT ResponseDestination(  
    [in] IDispatch* pdestResponse  
);
```

pdestResponse: A pointer to an IDispatch interface that contains an MSMQDestination object representing the set of queues to send the response messages to.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- Obtain the *QueueFormatName* from the pdestResponse instance state.
- The [Response Queue Format Name] property of the represented [Message] MUST be set to the obtained format name.

3.17.4.1.82 Destination (Opnum 88)

The **Destination** method is received by the server in an RPC_REQUEST packet. In response, the server returns an IDispatch interface pointer to an MSMQDestination object that represents 0 or more [Queue]s identified by the [Destination Queue Format Name] property of the represented [Message].

```
[propget] HRESULT Destination(  
    [out, retval] IDispatch** ppdestDestination  
);
```

ppdestDestination: A pointer to a pointer to an IDispatch interface that upon successful completion will contain an MSMQDestination object representing the set of queues to send the message to.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The server MUST create a new MSMQDestination object instance
- Set the *QueueFormatName* instance variable of the created MSMQDestination object to the value of the [Destination Queue Format Name] property of the represented [Message].
- The *ppdestDestination* output parameter MUST be set to a pointer to an IDispatch4 interface for the newly created MSMQDestination object

3.17.4.1.83 LookupId (Opnum 89)

The **LookupId** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [Lookup Identifier] property of the represented [Message].

```
[propget] HRESULT LookupId(  
    [out, retval] VARIANT* pvarLookupId  
);
```

pvarLookupId: A pointer to a VARIANT containing a BSTR which contains the 64-bit lookup identifier for the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pvarLookupId* output parameter MUST be set to the value of the [Lookup Identifier] property of the represented [Message].

3.17.4.1.84 IsAuthenticated2 (Opnum 90)

The **IsAuthenticated2** method is received by the server in an RPC_REQUEST packet. In response, the server returns a Boolean flag indicating if the message was authenticated by the [Queue Manager] that received the [Message] This provides identical behavior to the method [IMSMQMessage4::IsAuthenticated \(section 3.17.4.1.6\)](#).

```
[propget] HRESULT IsAuthenticated2(  
    [out, retval] VARIANT_BOOL* pisAuthenticated  
);
```

pisAuthenticated: A pointer to a VARIANT_BOOL that specifies whether or not the message was authenticated.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the [Authentication Level] property of the represented [Message] is MQMSG_AUTHENTICATION_NOT_REQUESTED (0x00000000)
 - The *pisAuthenticated* output parameter MUST be set to VARIANT_FALSE (0x0000).

- Else
 - The *pisAuthenticated* output parameter MUST be set to VARIANT_TRUE (0xffff).

3.17.4.1.85 IsFirstInTransaction2 (Opnum 91)

The **IsFirstInTransaction2** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IsFirstInTransaction] property of the represented [Message].

```
[propget] HRESULT IsFirstInTransaction2(
    [out, retval] VARIANT_BOOL* pisFirstInXact
);
```

pisFirstInXact: A pointer to a VARIANT_BOOL that specifies whether the message was the first sent by the transaction.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pisFirstInXact* output parameter MUST be set to the value of the [IsFirstInTransaction] property of the represented [Message]

3.17.4.1.86 IsLastInTransaction2 (Opnum 92)

The **IsLastInTransaction2** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [IsLastInTransaction] property of the represented [Message].

```
[propget] HRESULT IsLastInTransaction2(
    [out, retval] VARIANT_BOOL* pisLastInXact
);
```

pisLastInXact: A pointer to a VARIANT_BOOL that specifies whether the message was the last sent by the transaction.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pisLastInXact* output parameter MUST be set to the value of the [IsLastInTransaction] property of the represented [Message]

3.17.4.1.87 AttachCurrentSecurityContext2 (Opnum 93)

The **AttachCurrentSecurityContext2** method requests that the server caches the relevant information required to sign a message on behalf of a user; including the [Sender Identifier], [Sender Certificate] and [Symmetric Key]. This method is provided purely as an optimization to allow the client to reduce the number of round-trips to the server to specify the required information to sign and encrypt a message. If any of these values are not specified by the client, the server will query the information from the [Directory]. The [Sender Identifier], [Sender Certificate] and [Symmetric Key] property values MUST NOT be updated with the respective values if a [Directory] lookup is performed by the server.

```
HRESULT AttachCurrentSecurityContext2();
```

This method has no parameters.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- If the [Sender Identifier] property of the represented [Message] equals NULL
 - Obtain the sender identifier of the calling user and set the *SenderIdentifierCache* instance variable to the sender identifier.
- Else
 - Set the *SenderIdentifierCache* instance variable to the value of the [Sender Identifier] property of the represented [Message].
- If the [Sender Certificate] property of the represented [Message] equals NULL
 - Set the [Sender Identifier] property of the represented [Message] to the **internal certificate** from the internal certificate store for the user identified by the [Sender Identifier] property of the represented [Message]
- Else
 - Lookup the certificate in the personal certificate store using the [Sender Certificate] property value.
 - If found
 - Set the *SenderCertificateCache* instance variable to the value of the [Sender Certificate] property of the represented [Message].
 - Else
 - The server MUST return an error and take no further action.
- If the [Symmetric Key] property of the represented [Message] is NOT NULL
 - Set the *SymmetricKeyCache* instance variable to the value of the [Symmetric Key] property of the represented [Message].

3.17.4.1.88 SoapEnvelope (Opnum 94)

The **SoapEnvelope** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [SOAP Envelope] property of the represented [Message].

```
[propget] HRESULT SoapEnvelope(  
    [out, retval] BSTR* pbstrSoapEnvelope  
);
```

pbstrSoapEnvelope: A pointer to a BSTR containing a string of Unicode characters that contains the SOAP envelope.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pbstrSoapEnvelope* output parameter MUST be set to the value of the [SOAP Envelope] property of the represented [Message].

3.17.4.1.89 CompoundMessage (Opnum 95)

The **CompoundMessage** method is received by the server in an RPC_REQUEST packet. In response, the server returns the [SOAP Compound Message] property of the represented [Message].

```
[propget] HRESULT CompoundMessage(  
    [out, retval] VARIANT* pvarCompoundMessage  
);
```

pvarCompoundMessage: A pointer to a VARIANT that contains an array of BYTES (VT_ARRAY | VT_UI1) which contains the entire contents of the SRMP message including the SOAP envelope and all SOAP attachments.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *pvarCompoundMessage* output parameter MUST be set to the value of the [SOAP Compound Message] property of the represented [Message].

3.17.4.1.90 SoapHeader (Opnum 96)

The **SoapHeader** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *SOAPHeader* instance variable.

```
[propput] HRESULT SoapHeader(  
    [in] BSTR bstrSoapHeader  
);
```

bstrSoapHeader: A BSTR that contains a string representation of the XML elements that will be appended to the end of the SOAP header within the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *SOAPHeader* instance variable MUST be set to the value in the *bstrSoapHeader* input parameter.

3.17.4.1.91 SoapBody (Opnum 97)

The **SoapBody** method is received by the server in an RPC_REQUEST packet. In response, the server sets the *SOAPBody* instance variable.

```
[propput] HRESULT SoapBody(  
    [in] BSTR bstrSoapBody  
);
```

bstrSoapBody: A BSTR that contains a string representation of the XML elements that will be appended to the end of the SOAP body within the message.

Return Values: The method MUST return S_OK (0x00000000) on success, or an implementation-specific error HRESULT on failure.

When processing this call, the server MUST follow these guidelines:

- The *SOAPBody* instance variable MUST be set to the value in the *bstrSoapBody* input parameter.

3.17.5 Timer Events

There are no timer events for this coclass.

3.17.6 Other Local Events

There are no other local events for this coclass.

4 Protocol Examples

There are no specific examples provided for this protocol.

5 Security

5.1 Security Considerations for Implementers

The protocol defined in this document provides an abstract interface to a message queuing system and related services. Server implementations of this protocol are responsible for rendering the security models of the underlying systems and services within the abstract interfaces defined by the protocol. Consequently, clients of this protocol are subject to the security models imposed by underlying messaging systems and services.

5.2 Index of Security Parameters

No security parameters are defined for this protocol.

6 Appendix A: Full IDL

For ease of implementation, the full IDL is provided below, where "ms-dtyp.idl" refers to the IDL found in [\[MS-DTYP\] Appendix A](#) and "ms-oadt.idl" refers to the IDL found in [\[MS-OADT\] Appendix A](#).

The syntax uses the IDL syntax extensions defined in [\[MS-RPCE\] sections 2.2.4 and 3.1.1.5.1](#). For example, as noted in [\[MS-RPCE\] section 2.2.4.8](#), a `pointer_default` declaration is not required, and `pointer_default(unique)` is assumed.

```
import "ms-dtyp.idl";
import "ms-oadt.idl";

typedef struct BOID {
    unsigned char rgb[16];
} BOID;

typedef struct XACTTRANSINFO {
    BOID    uow;
    LONG    isoLevel;
    ULONG    isoFlags;
    DWORD    grfTCSupported;
    DWORD    grfRMSupported;
    DWORD    grfTCSupportedRetaining;
    DWORD    grfRMSupportedRetaining;
} XACTTRANSINFO;

[
    object,
    uuid(0fb15084-af41-11ce-bd2b-204c4f4f5020),
    pointer_default(unique),
]
interface ITransaction : IUnknown {
    HRESULT Commit(
        [in] short    fRetaining,
        [in] DWORD    grfTC,
        [in] DWORD    grfRM
    );

    HRESULT Abort(
        [in, unique] BOID* pboidReason,
        [in] short    fRetaining,
        [in] short    fAsync
    );

    HRESULT GetTransactionInfo
    (
        [out] XACTTRANSINFO* pInfo
    );
}

[
    object,
    uuid(B196B287-BAB4-101A-B69C-00AA00341D07),
    pointer_default(unique)
]
interface IEnumConnections : IUnknown
{
    typedef IEnumConnections * PENUMCONNECTIONS;
    typedef IEnumConnections * LPENUMCONNECTIONS;
    typedef struct tagCONNECTDATA {
        IUnknown *    pUnk;
        DWORD        dwCookie;
    } CONNECTDATA;
}
```

```

typedef struct tagCONNECTDATA * PCONNECTDATA;
typedef struct tagCONNECTDATA * LPCONNECTDATA;

HRESULT Next(
    [in] ULONG cConnections,
    [out, size is(cConnections), length is(*pcFetched)]
        LPCONNECTDATA rgcd,
    [out] ULONG * pcFetched
);
HRESULT Skip(
    [in] ULONG cConnections
);
HRESULT Reset(
    void
);
HRESULT Clone(
    [out] IEnumConnections ** ppEnum
);
}

interface IConnectionPointContainer; // forward declaration

[
    object,
    uuid(B196B286-BAB4-101A-B69C-00AA00341D07),
    pointer_default(unique)
]
interface IConnectionPoint : IUnknown
{
    typedef IConnectionPoint * PCONNECTIONPOINT;
    typedef IConnectionPoint * LPCONNECTIONPOINT;
    HRESULT GetConnectionInterface(
        [out] IID * pIID
    );
    HRESULT GetConnectionPointContainer(
        [out] IConnectionPointContainer ** ppCPC
    );
    HRESULT Advise(
        [in] IUnknown * pUnkSink,
        [out] DWORD * pdwCookie
    );
    HRESULT Unadvise(
        [in] DWORD dwCookie
    );
    HRESULT EnumConnections(
        [out] IEnumConnections ** ppEnum
    );
}

[
    object,
    uuid(B196B285-BAB4-101A-B69C-00AA00341D07),
    pointer default(unique)
]
interface IEnumConnectionPoints : IUnknown
{
    typedef IEnumConnectionPoints * PENUMCONNECTIONPOINTS;
    typedef IEnumConnectionPoints * LPENUMCONNECTIONPOINTS;
    HRESULT Next(
        [in] ULONG cConnections,
        [out, size is(cConnections), length is(*pcFetched)]
            LPCONNECTIONPOINT * ppCP,
        [out] ULONG * pcFetched
    );
    HRESULT Skip(

```

```

        [in] ULONG cConnections
    );
    HRESULT Reset(
        void
    );
    HRESULT Clone(
        [out] IEnumConnectionPoints ** ppEnum
    );
}

[
    object,
    uuid(B196B284-BAB4-101A-B69C-00AA00341D07),
    pointer_default(unique)
]
interface IConnectionPointContainer : IUnknown
{
    typedef IConnectionPointContainer * PCONNECTIONPOINTCONTAINER;
    typedef IConnectionPointContainer * LPCONNECTIONPOINTCONTAINER;
    HRESULT EnumConnectionPoints
    (
        [out] IEnumConnectionPoints ** ppEnum
    );
    HRESULT FindConnectionPoint
    (
        [in] REFIID riid,
        [out] IConnectionPoint ** ppCP
    );
}

// forwards defs
interface IMSMQQuery;
interface IMSMQQueueInfo;
interface IMSMQQueueInfo2;
interface IMSMQQueueInfo3;
interface IMSMQQueueInfo4;
interface IMSMQQueue;
interface IMSMQQueue2;
interface IMSMQQueue3;
interface IMSMQQueue4;
interface IMSMQMessage;
interface IMSMQQueueInfos;
interface IMSMQQueueInfos2;
interface IMSMQQueueInfos3;
interface IMSMQQueueInfos4;
interface IMSMQEvent;
interface IMSMQEvent2;
interface IMSMQEvent3;
interface IMSMQTransaction;
interface IMSMQCoordinatedTransactionDispenser;
interface IMSMQTransactionDispenser;
enum MQCALG {
    MQMSG_CALG_MD2          = 0x8001,
    MQMSG_CALG_MD4          = 0x8002,
    MQMSG_CALG_MD5          = 0x8003,
    MQMSG_CALG_SHA          = 0x8004,
    MQMSG_CALG_SHA1         = 0x8004,
    MQMSG_CALG_MAC          = 0x8005,
    MQMSG_CALG_RSA_SIGN     = 0x2400,
    MQMSG_CALG_DSS_SIGN     = 0x2200,
    MQMSG_CALG_RSA_KEYX     = 0xA400,
    MQMSG_CALG_DES          = 0x6601,
    MQMSG_CALG_RC2          = 0x6602,
    MQMSG_CALG_RC4          = 0x6801,

```

```

    MQMSG CALG SEAL          = 0x6802
};
enum MQTRANSACTION {
    MQ NO TRANSACTION        = 0,
    MQ MTS TRANSACTION       = 1,
    MQ XA TRANSACTION        = 2,
    MQ SINGLE MESSAGE        = 3
};
enum RELOPS {
    REL NOP      = 0,
    REL EQ,
    REL NEQ,
    REL LT,
    REL_GT,
    REL_LE,
    REL GE
};
enum MQCERT REGISTER {
    MQCERT REGISTER ALWAYS      = 1,
    MQCERT_REGISTER_IF_NOT_EXIST = 2
};
enum MQMSGCURSOR {
    MQMSG FIRST = 0,
    MQMSG CURRENT = 1,
    MQMSG_NEXT = 2
};
enum MQMSGCLASS {
    MQMSG CLASS NORMAL                = 0x0,
    MQMSG CLASS REPORT                = 0x1,
    MQMSG CLASS ACK REACH QUEUE       = 0x2,
    MQMSG CLASS ACK RECEIVE           = 0x4000,
    MQMSG_CLASS_NACK_BAD_DST_Q        = 0x8000,
    MQMSG_CLASS_NACK_PURGED           = 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 COULD NOT ENCRYPT  = 0x8008,
    MQMSG CLASS NACK NOT TRANSACTIONAL Q = 0x8009,
    MQMSG CLASS NACK NOT TRANSACTIONAL MSG = 0x800a,
    MQMSG_CLASS_NACK_UNSUPPORTED_CRYPTO_PROVIDER = 0x800b,
    MQMSG_CLASS_NACK_SOURCE_COMPUTER_GUID_CHANGED = 0x800c,
    MQMSG CLASS NACK Q DELETED         = 0xc000,
    MQMSG CLASS NACK Q PURGED          = 0xc001,
    MQMSG CLASS NACK RECEIVE TIMEOUT   = 0xc002,
    MQMSG CLASS NACK RECEIVE TIMEOUT AT SENDER = 0xc003
};
enum MQMSGDELIVERY {
    MQMSG DELIVERY EXPRESS            = 0,
    MQMSG DELIVERY RECOVERABLE       = 1
};
enum MQMSGACKNOWLEDGEMENT {
    MQMSG_ACKNOWLEDGMENT_NONE        = 0x00,
    MQMSG_ACKNOWLEDGMENT_POS_ARRIVAL  = 0x01,
    MQMSG_ACKNOWLEDGMENT_POS_RECEIVE  = 0x02,
    MQMSG_ACKNOWLEDGMENT_NEG_ARRIVAL  = 0x04,
    MQMSG_ACKNOWLEDGMENT_NEG_RECEIVE  = 0x08,
    MQMSG_ACKNOWLEDGMENT_NACK_REACH_QUEUE = 0x04,
    MQMSG_ACKNOWLEDGMENT_FULL_REACH_QUEUE = 0x05,
    MQMSG_ACKNOWLEDGMENT_NACK_RECEIVE  = 0x0c,
    MQMSG_ACKNOWLEDGMENT_FULL_RECEIVE  = 0x0e
};
enum MQMSGJOURNAL {

```

```

MQMSG JOURNAL NONE      = 0,
MQMSG DEADLETTER        = 1,
MQMSG_JOURNAL           = 2
};
enum MQMSGTRACE {
MQMSG TRACE NONE        = 0,
MQMSG SEND ROUTE TO REPORT QUEUE = 1
};
enum MQMSGSENDERIDTYPE {
MQMSG SENDERID TYPE NONE = 0,
MQMSG SENDERID TYPE SID  = 1
};
enum MQMSGPRIVLEVEL {
MQMSG_PRIV_LEVEL_NONE      = 0,
MQMSG_PRIV_LEVEL_BODY_BASE = 1,
MQMSG_PRIV_LEVEL_BODY_ENHANCED = 3
};
enum MQMSGAUTHLEVEL {
MQMSG AUTH LEVEL NONE      = 0,
MQMSG_AUTH_LEVEL_ALWAYS    = 1,
MQMSG_AUTH_LEVEL_MSMQ10     = 2,
MQMSG_AUTH_LEVEL_SIG10     = 2,
MQMSG_AUTH_LEVEL_MSMQ20     = 4,
MQMSG_AUTH_LEVEL_SIG20     = 4,
MQMSG_AUTH_LEVEL_SIG30     = 8
};
enum MQMSGIDSIZE {
MQMSG MSGID SIZE           = 20,
MQMSG CORRELATIONID SIZE   = 20,
MQMSG XACTID SIZE          = 20
};
enum MQMSGMAX {
MQ_MAX_MSG_LABEL_LEN      = 249,
};
enum MQMSGAUTHENTICATION {
MQMSG AUTHENTICATION NOT REQUESTED = 0,
MQMSG AUTHENTICATION REQUESTED     = 1,
MQMSG_AUTHENTICATED_SIG10          = 1,
MQMSG_AUTHENTICATION_REQUESTED_EX  = 3,
MQMSG_AUTHENTICATED_SIG20          = 3,
MQMSG_AUTHENTICATED_SIG30          = 5,
MQMSG_AUTHENTICATED_SIGXML         = 9
};
enum MQSHARE {
MQ_DENY_NONE                = 0,
MQ_DENY_RECEIVE_SHARE       = 1
};
enum MQACCESS {
MQ_RECEIVE_ACCESS           = 1,
MQ_SEND_ACCESS              = 2,
MQ_PEEK_ACCESS              = 0x20,
MQ_ADMIN_ACCESS             = 0x80
};
enum MQJOURNAL {
MQ_JOURNAL_NONE             = 0,
MQ_JOURNAL                  = 1
};
enum MQTRANSACTIONAL {
MQ_TRANSACTIONAL_NONE       = 0,
MQ_TRANSACTIONAL            = 1
};
enum MQAUTHENTICATE {
MQ_AUTHENTICATE_NONE        = 0,
MQ_AUTHENTICATE             = 1
};

```

```

enum MQPRIVLEVEL {
    MQ_PRIV_LEVEL_NONE        = 0,
    MQ_PRIV_LEVEL_OPTIONAL    = 1,
    MQ_PRIV_LEVEL_BODY        = 2
};

enum MQPRIORITY {
    MQ_MIN_PRIORITY           = 0,
    MQ_MAX_PRIORITY           = 7
};

enum MQMAX {
    MQ_MAX_Q_NAME_LEN        = 124,
    MQ_MAX_Q_LABEL_LEN       = 124
};

enum QUEUE_TYPE {
    MQ_TYPE_PUBLIC,
    MQ_TYPE_PRIVATE,
    MQ_TYPE_MACHINE,
    MQ_TYPE_CONNECTOR,
    MQ_TYPE_MULTICAST
};

enum FOREIGN_STATUS {
    MQ_STATUS_FOREIGN,
    MQ_STATUS_NOT_FOREIGN,
    MQ_STATUS_UNKNOWN
};

enum XACT_STATUS {
    MQ_XACT_STATUS_XACT,
    MQ_XACT_STATUS_NOT_XACT,
    MQ_XACT_STATUS_UNKNOWN
};

enum QUEUE_STATE{
    MQ_QUEUE_STATE_LOCAL_CONNECTION,
    MQ_QUEUE_STATE_DISCONNECTED,
    MQ_QUEUE_STATE_WAITING,
    MQ_QUEUE_STATE_NEEDVALIDATE,
    MQ_QUEUE_STATE_ONHOLD,
    MQ_QUEUE_STATE_NONACTIVE,
    MQ_QUEUE_STATE_CONNECTED,
    MQ_QUEUE_STATE_DISCONNECTING,
    MQ_QUEUE_STATE_LOCKED
};

enum MQDEFAULT {
    DEFAULT_M_PRIORITY         = 3,
    DEFAULT_M_DELIVERY         = 0,
    DEFAULT_M_ACKNOWLEDGE      = 0,
    DEFAULT_M_JOURNAL          = 0,
    DEFAULT_M_APPSPECIFIC      = 0,
    DEFAULT_M_PRIV_LEVEL       = 0,
    DEFAULT_M_AUTH_LEVEL       = 0,
    DEFAULT_M_SENDERID_TYPE    = 1,
    DEFAULT_Q_JOURNAL          = 0,
    DEFAULT_Q_BASEPRIORITY     = 0,
    DEFAULT_Q_QUOTA             = 0xFFFFFFFF,
    DEFAULT_Q_JOURNAL_QUOTA    = 0xFFFFFFFF,
    DEFAULT_Q_TRANSACTION      = 0,
    DEFAULT_Q_AUTHENTICATE     = 0,
    DEFAULT_Q_PRIV_LEVEL       = 1,
    DEFAULT_M_LOOKUPID         = 0,
};

enum MQERROR {
    MQ_ERROR                   = 0xC00E0001,
    MQ_ERROR_PROPERTY          = 0xC00E0002,
    MQ_ERROR_QUEUE_NOT_FOUND   = 0xC00E0003,
    MQ_ERROR_QUEUE_NOT_ACTIVE  = 0xC00E0004,
    MQ_ERROR_QUEUE_EXISTS      = 0xC00E0005,
};

```


MQ_ERROR_INVALID_PARAMETER	= 0xC00E0006,
MQ_ERROR_INVALID_HANDLE	= 0xC00E0007,
MQ_ERROR_OPERATION_CANCELLED	= 0xC00E0008,
MQ_ERROR_SHARING_VIOLATION	= 0xC00E0009,
MQ_ERROR_SERVICE_NOT_AVAILABLE	= 0xC00E000B,
MQ_ERROR_MACHINE_NOT_FOUND	= 0xC00E000D,
MQ_ERROR_ILLEGAL_SORT	= 0xC00E0010,
MQ_ERROR_ILLEGAL_USER	= 0xC00E0011,
MQ_ERROR_NO_DS	= 0xC00E0013,
MQ_ERROR_ILLEGAL_QUEUE_PATHNAME	= 0xC00E0014,
MQ_ERROR_ILLEGAL_PROPERTY_VALUE	= 0xC00E0018,
MQ_ERROR_ILLEGAL_PROPERTY_VT	= 0xC00E0019,
MQ_ERROR_BUFFER_OVERFLOW	= 0xC00E001A,
MQ_ERROR_IO_TIMEOUT	= 0xC00E001B,
MQ_ERROR_ILLEGAL_CURSOR_ACTION	= 0xC00E001C,
MQ_ERROR_MESSAGE_ALREADY_RECEIVED	= 0xC00E001D,
MQ_ERROR_ILLEGAL_FORMATNAME	= 0xC00E001E,
MQ_ERROR_FORMATNAME_BUFFER_TOO_SMALL	= 0xC00E001F,
MQ_ERROR_UNSUPPORTED_FORMATNAME_OPERATION	= 0xC00E0020,
MQ_ERROR_ILLEGAL_SECURITY_DESCRIPTOR	= 0xC00E0021,
MQ_ERROR_SENDERID_BUFFER_TOO_SMALL	= 0xC00E0022,
MQ_ERROR_SECURITY_DESCRIPTOR_TOO_SMALL	= 0xC00E0023,
MQ_ERROR_CANNOT_IMPERSONATE_CLIENT	= 0xC00E0024,
MQ_ERROR_ACCESS_DENIED	= 0xC00E0025,
MQ_ERROR_PRIVILEGE_NOT_HELD	= 0xC00E0026,
MQ_ERROR_INSUFFICIENT_RESOURCES	= 0xC00E0027,
MQ_ERROR_USER_BUFFER_TOO_SMALL	= 0xC00E0028,
MQ_ERROR_MESSAGE_STORAGE_FAILED	= 0xC00E002A,
MQ_ERROR_SENDER_CERT_BUFFER_TOO_SMALL	= 0xC00E002B,
MQ_ERROR_INVALID_CERTIFICATE	= 0xC00E002C,
MQ_ERROR_CORRUPTED_INTERNAL_CERTIFICATE	= 0xC00E002D,
MQ_ERROR_INTERNAL_USER_CERT_EXISTS	= 0xC00E002E,
MQ_ERROR_NO_INTERNAL_USER_CERT	= 0xC00E002F,
MQ_ERROR_CORRUPTED_SECURITY_DATA	= 0xC00E0030,
MQ_ERROR_CORRUPTED_PERSONAL_CERT_STORE	= 0xC00E0031,
MQ_ERROR_COMPUTER_DOES_NOT_SUPPORT_ENCRYPTION	= 0xC00E0033,
MQ_ERROR_BAD_SECURITY_CONTEXT	= 0xC00E0035,
MQ_ERROR_COULD_NOT_GET_USER_SID	= 0xC00E0036,
MQ_ERROR_COULD_NOT_GET_ACCOUNT_INFO	= 0xC00E0037,
MQ_ERROR_ILLEGAL_MQ_COLUMNS	= 0xC00E0038,
MQ_ERROR_ILLEGAL_PROPID	= 0xC00E0039,
MQ_ERROR_ILLEGAL_RELATION	= 0xC00E003A,
MQ_ERROR_ILLEGAL_PROPERTY_SIZE	= 0xC00E003B,
MQ_ERROR_ILLEGAL_RESTRICTION_PROPID	= 0xC00E003C,
MQ_ERROR_ILLEGAL_MQ_QUEUE_PROPS	= 0xC00E003D,
MQ_ERROR_PROPERTY_NOT_ALLOWED	= 0xC00E003E,
MQ_ERROR_INSUFFICIENT_PROPERTIES	= 0xC00E003F,
MQ_ERROR_MACHINE_EXISTS	= 0xC00E0040,
MQ_ERROR_ILLEGAL_MQ_QUEUE_PROPS	= 0xC00E0041,
MQ_ERROR_DS_IS_FULL	= 0xC00E0042L,
MQ_ERROR_DS_ERROR	= 0xC00E0043,
MQ_ERROR_INVALID_OWNER	= 0xC00E0044,
MQ_ERROR_UNSUPPORTED_ACCESS_MODE	= 0xC00E0045,
MQ_ERROR_RESULT_BUFFER_TOO_SMALL	= 0xC00E0046,
MQ_ERROR_DELETE_CN_IN_USE	= 0xC00E0048L,
MQ_ERROR_NO_RESPONSE_FROM_OBJECT_SERVER	= 0xC00E0049,
MQ_ERROR_OBJECT_SERVER_NOT_AVAILABLE	= 0xC00E004A,
MQ_ERROR_QUEUE_NOT_AVAILABLE	= 0xC00E004B,
MQ_ERROR_DTC_CONNECT	= 0xC00E004C,
MQ_ERROR_TRANSACTION_IMPORT	= 0xC00E004E,
MQ_ERROR_TRANSACTION_USAGE	= 0xC00E0050,
MQ_ERROR_TRANSACTION_SEQUENCE	= 0xC00E0051,
MQ_ERROR_MISSING_CONNECTOR_TYPE	= 0xC00E0055,
MQ_ERROR_STALE_HANDLE	= 0xC00E0056,
MQ_ERROR_TRANSACTION_ENLIST	= 0xC00E0058,

MQ_ERROR_QUEUE_DELETED	= 0xC00E005A,
MQ_ERROR_ILLEGAL_CONTEXT	= 0xC00E005B,
MQ_ERROR_ILLEGAL_SORT_PROPID	= 0xC00E005C,
MQ_ERROR_LABEL_TOO_LONG	= 0xC00E005D,
MQ_ERROR_LABEL_BUFFER_TOO_SMALL	= 0xC00E005E,
MQ_ERROR_MQIS_SERVER_EMPTY	= 0xC00E005F,
MQ_ERROR_MQIS_READONLY_MODE	= 0xC00E0060,
MQ_ERROR_SYMM_KEY_BUFFER_TOO_SMALL	= 0xC00E0061,
MQ_ERROR_SIGNATURE_BUFFER_TOO_SMALL	= 0xC00E0062,
MQ_ERROR_PROV_NAME_BUFFER_TOO_SMALL	= 0xC00E0063,
MQ_ERROR_ILLEGAL_OPERATION	= 0xC00E0064,
MQ_ERROR_WRITE_NOT_ALLOWED	= 0xC00E0065,
MQ_ERROR_WKS_CANT_SERVE_CLIENT	= 0xC00E0066,
MQ_ERROR_DEPEND_WKS_LICENSE_OVERFLOW	= 0xC00E0067,
MQ_ERROR_CORRUPTED_QUEUE_WAS_DELETED	= 0xC00E0068,
MQ_ERROR_REMOTE_MACHINE_NOT_AVAILABLE	= 0xC00E0069,
MQ_ERROR_UNSUPPORTED_OPERATION	= 0xC00E006A,
MQ_ERROR_ENCRYPTION_PROVIDER_NOT_SUPPORTED	= 0xC00E006B,
MQ_ERROR_CANNOT_SET_CRYPTO_SEC_DESCR	= 0xC00E006C,
MQ_ERROR_CERTIFICATE_NOT_PROVIDED	= 0xC00E006D,
MQ_ERROR_Q_DNS_PROPERTY_NOT_SUPPORTED	= 0xC00E006E,
MQ_ERROR_CANT_CREATE_CERT_STORE	= 0xC00E006F,
MQ_ERROR_CANNOT_CREATE_CERT_STORE	= 0xC00E006F,
MQ_ERROR_CANT_OPEN_CERT_STORE	= 0xC00E0070,
MQ_ERROR_CANNOT_OPEN_CERT_STORE	= 0xC00E0070,
MQ_ERROR_ILLEGAL_ENTERPRISE_OPERATION	= 0xC00E0071,
MQ_ERROR_CANNOT_GRANT_ADD_GUID	= 0xC00E0072,
MQ_ERROR_CANNOT_LOAD MSMQOCM	= 0xC00E0073,
MQ_ERROR_NO_ENTRY_POINT MSMQOCM	= 0xC00E0074,
MQ_ERROR_NO_MSMQ_SERVERS_ON_DC	= 0xC00E0075,
MQ_ERROR_CANNOT_JOIN_DOMAIN	= 0xC00E0076,
MQ_ERROR_CANNOT_CREATE_ON_GC	= 0xC00E0077,
MQ_ERROR_GUID_NOT_MATCHING	= 0xC00E0078,
MQ_ERROR_PUBLIC_KEY_NOT_FOUND	= 0xC00E0079,
MQ_ERROR_PUBLIC_KEY_DOES_NOT_EXIST	= 0xC00E007A,
MQ_ERROR_ILLEGAL MQPRIVATEPROPS	= 0xC00E007B,
MQ_ERROR_NO_GC_IN_DOMAIN	= 0xC00E007C,
MQ_ERROR_NO_MSMQ_SERVERS_ON_GC	= 0xC00E007D,
MQ_ERROR_CANNOT_GET_DN	= 0xC00E007E,
MQ_ERROR_CANNOT_HASH_DATA_EX	= 0xC00E007F,
MQ_ERROR_CANNOT_SIGN_DATA_EX	= 0xC00E0080,
MQ_ERROR_CANNOT_CREATE_HASH_EX	= 0xC00E0081,
MQ_ERROR_FAIL_VERIFY_SIGNATURE_EX	= 0xC00E0082,
MQ_ERROR_CANNOT_DELETE_PSC_OBJECTS	= 0xC00E0083,
MQ_ERROR_NO MQUSER OU	= 0xC00E0084,
MQ_ERROR_CANNOT_LOAD MQAD	= 0xC00E0085,
MQ_ERROR_CANNOT_LOAD MQDSSRV	= 0xC00E0086,
MQ_ERROR_PROPERTIES_CONFLICT	= 0xC00E0087,
MQ_ERROR_MESSAGE_NOT_FOUND	= 0xC00E0088,
MQ_ERROR_CANT_RESOLVE_SITES	= 0xC00E0089,
MQ_ERROR_NOT_SUPPORTED_BY_DEPENDENT_CLIENTS	= 0xC00E008A,
MQ_ERROR_OPERATION_NOT_SUPPORTED_BY_REMOTE_COMPUTER =	
	0xC00E008B,
MQ_ERROR_NOT_A_CORRECT_OBJECT_CLASS	= 0xC00E008C,
MQ_ERROR_MULTI_SORT_KEYS	= 0xC00E008D,
MQ_ERROR_GC_NEEDED	= 0xC00E008E,
MQ_ERROR_DS_BIND_ROOT_FOREST	= 0xC00E008F,
MQ_ERROR_DS_LOCAL_USER	= 0xC00E0090,
MQ_ERROR_Q_ADS_PROPERTY_NOT_SUPPORTED	= 0xC00E0091,
MQ_ERROR_BAD_XML_FORMAT	= 0xC00E0092,
MQ_ERROR_UNSUPPORTED_CLASS	= 0xC00E0093,
MQ_ERROR_UNINITIALIZED_OBJECT	= 0xC00E0094,
MQ_ERROR_CANNOT_CREATE_PSC_OBJECTS	= 0xC00E0095,
MQ_ERROR_CANNOT_UPDATE_PSC_OBJECTS	= 0xC00E0096

};

```

enum MQWARNING {
    MQ_INFORMATION_PROPERTY                = 0x400E0001,
    MQ_INFORMATION_ILLEGAL_PROPERTY        = 0x400E0002,
    MQ_INFORMATION_PROPERTY_IGNORED        = 0x400E0003,
    MQ_INFORMATION_UNSUPPORTED_PROPERTY     = 0x400E0004,
    MQ_INFORMATION_DUPLICATE_PROPERTY      = 0x400E0005,
    MQ_INFORMATION_OPERATION_PENDING       = 0x400E0006,
    MQ_INFORMATION_FORMATNAME_BUFFER_TOO_SMALL = 0x400E0009,
    MQ_INFORMATION_INTERNAL_USER_CERT_EXIST = 0x400E000A,
    MQ_INFORMATION_OWNER_IGNORED           = 0x400E000B,
};

[
    uuid(D7D6E072-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, nonextensible, odl
]
interface IMSMQQuery : IDispatch {
    HRESULT LookupQueue(
        [in, optional] VARIANT *QueueGuid,
        [in, optional] VARIANT *ServiceTypeGuid,
        [in, optional] VARIANT *Label,
        [in, optional] VARIANT *CreateTime,
        [in, optional] VARIANT *ModifyTime,
        [in, optional] VARIANT *RelServiceType,
        [in, optional] VARIANT *RelLabel,
        [in, optional] VARIANT *RelCreateTime,
        [in, optional] VARIANT *RelModifyTime,
        [out, retval] IMSMQQueueInfos **ppqinfos
    );
};

[
    uuid(eba96b0e-2168-11d3-898c-00e02c074f6b),
    hidden, dual, nonextensible, odl
]
interface IMSMQQuery2 : IDispatch {
    HRESULT LookupQueue(
        [in, optional] VARIANT *QueueGuid,
        [in, optional] VARIANT *ServiceTypeGuid,
        [in, optional] VARIANT *Label,
        [in, optional] VARIANT *CreateTime,
        [in, optional] VARIANT *ModifyTime,
        [in, optional] VARIANT *RelServiceType,
        [in, optional] VARIANT *RelLabel,
        [in, optional] VARIANT *RelCreateTime,
        [in, optional] VARIANT *RelModifyTime,
        [out, retval] IMSMQQueueInfos2 **ppqinfos
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};

[
    uuid(eba96b19-2168-11d3-898c-00e02c074f6b),
    hidden, dual, nonextensible, odl
]
interface IMSMQQuery3 : IDispatch {
    HRESULT LookupQueue v2(
        [in, optional] VARIANT *QueueGuid,
        [in, optional] VARIANT *ServiceTypeGuid,
        [in, optional] VARIANT *Label,
        [in, optional] VARIANT *CreateTime,
        [in, optional] VARIANT *ModifyTime,
        [in, optional] VARIANT *RelServiceType,
        [in, optional] VARIANT *RelLabel,

```

```

        [in, optional] VARIANT *RelCreateTime,
        [in, optional] VARIANT *RelModifyTime,
        [out, retval] IMSMQQueueInfos3 **ppqinfos
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
    HRESULT LookupQueue(
        [in, optional] VARIANT *QueueGuid,
        [in, optional] VARIANT *ServiceTypeGuid,
        [in, optional] VARIANT *Label,
        [in, optional] VARIANT *CreateTime,
        [in, optional] VARIANT *ModifyTime,
        [in, optional] VARIANT *RelServiceType,
        [in, optional] VARIANT *RelLabel,
        [in, optional] VARIANT *RelCreateTime,
        [in, optional] VARIANT *RelModifyTime,
        [in, optional] VARIANT *MulticastAddress,
        [in, optional] VARIANT *RelMulticastAddress,
        [out, retval] IMSMQQueueInfos3 **ppqinfos
    );
};

[
    uuid(eba96b24-2168-11d3-898c-00e02c074f6b),
    hidden, dual, nonextensible, odl
]
interface IMSMQQuery4 : IDispatch {
    HRESULT LookupQueue v2(
        [in, optional] VARIANT *QueueGuid,
        [in, optional] VARIANT *ServiceTypeGuid,
        [in, optional] VARIANT *Label,
        [in, optional] VARIANT *CreateTime,
        [in, optional] VARIANT *ModifyTime,
        [in, optional] VARIANT *RelServiceType,
        [in, optional] VARIANT *RelLabel,
        [in, optional] VARIANT *RelCreateTime,
        [in, optional] VARIANT *RelModifyTime,
        [out, retval] IMSMQQueueInfos4 **ppqinfos
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
    HRESULT LookupQueue(
        [in, optional] VARIANT *QueueGuid,
        [in, optional] VARIANT *ServiceTypeGuid,
        [in, optional] VARIANT *Label,
        [in, optional] VARIANT *CreateTime,
        [in, optional] VARIANT *ModifyTime,
        [in, optional] VARIANT *RelServiceType,
        [in, optional] VARIANT *RelLabel,
        [in, optional] VARIANT *RelCreateTime,
        [in, optional] VARIANT *RelModifyTime,
        [in, optional] VARIANT *MulticastAddress,
        [in, optional] VARIANT *RelMulticastAddress,
        [out, retval] IMSMQQueueInfos4 **ppqinfos
    );
};

[
    uuid(D7D6E073-DCCD-11d0-AA4B-0060970DEBAE),
]
coclass MSMQQuery {
    interface IMSMQQuery;
    interface IMSMQQuery2;

```

```

interface IMSMQQuery3;
[default] interface IMSMQQuery4;
};

[
    uuid(D7D6E074-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, odl
]
interface IMSMQMessage : IDispatch {
    [propget] HRESULT Class([out, retval] long *plClass);
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT AuthLevel([out, retval] long *plAuthLevel);
    [propput] HRESULT AuthLevel([in] long lAuthLevel);
    [propget] HRESULT IsAuthenticated(
        [out, retval] short *pisAuthenticated
    );
    [propget] HRESULT Delivery([out, retval] long *plDelivery);
    [propput] HRESULT Delivery([in] long lDelivery);
    [propget] HRESULT Trace([out, retval] long *plTrace);
    [propput] HRESULT Trace([in] long lTrace);
    [propget] HRESULT Priority([out, retval] long *plPriority);
    [propput] HRESULT Priority([in] long lPriority);
    [propget] HRESULT Journal([out, retval] long *plJournal);
    [propput] HRESULT Journal([in] long lJournal);
    [propget] HRESULT ResponseQueueInfo(
        [out, retval] IMSMQQueueInfo **ppqinfoResponse
    );
    [propputref] HRESULT ResponseQueueInfo(
        [in] IMSMQQueueInfo *pqinfoResponse
    );
    [propget] HRESULT AppSpecific([out, retval] long *plAppSpecific);
    [propput] HRESULT AppSpecific([in] long lAppSpecific);
    [propget] HRESULT SourceMachineGuid(
        [out, retval] BSTR *pbstrGuidSrcMachine
    );
    [propget] HRESULT BodyLength([out, retval] long *pcbBody);
    [propget] HRESULT Body([out, retval] VARIANT *pvarBody);
    [propput] HRESULT Body([in] VARIANT varBody);
    [propget] HRESULT AdminQueueInfo(
        [out, retval] IMSMQQueueInfo **ppqinfoAdmin
    );
    [propputref] HRESULT AdminQueueInfo([in] IMSMQQueueInfo *pqinfoAdmin);
    [propget] HRESULT Id([out, retval] VARIANT *pvarMsgId);
    [propget] HRESULT CorrelationId([out, retval] VARIANT *pvarMsgId);
    [propput] HRESULT CorrelationId([in] VARIANT varMsgId);
    [propget] HRESULT Ack([out, retval] long *plAck);
    [propput] HRESULT Ack([in] long lAck);
    [propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
    [propput] HRESULT Label([in] BSTR bstrLabel);
    [propget] HRESULT MaxTimeToReachQueue(
        [out, retval] long *plMaxTimeToReachQueue
    );
    [propput] HRESULT MaxTimeToReachQueue([in] long lMaxTimeToReachQueue);
    [propget] HRESULT MaxTimeToReceive(
        [out, retval] long *plMaxTimeToReceive
    );
    [propput] HRESULT MaxTimeToReceive([in] long lMaxTimeToReceive);
    [propget] HRESULT HashAlgorithm([out, retval] long *plHashAlg);
    [propput] HRESULT HashAlgorithm([in] long lHashAlg);
    [propget] HRESULT EncryptAlgorithm([out, retval] long *plEncryptAlg);
    [propput] HRESULT EncryptAlgorithm([in] long lEncryptAlg);
    [propget] HRESULT SentTime([out, retval] VARIANT *pvarSentTime);
    [propget] HRESULT ArrivedTime([out, retval] VARIANT *plArrivedTime);
    [propget] HRESULT DestinationQueueInfo(

```

```

        [out, retval] IMSMQQueueInfo **ppqinfoDest
    );
    [propget] HRESULT SenderCertificate(
        [out, retval] VARIANT *pvarSenderCert
    );
    [propput] HRESULT SenderCertificate([in] VARIANT varSenderCert);
    [propget] HRESULT SenderId([out, retval] VARIANT *pvarSenderId);
    [propput] HRESULT SenderIdType([out, retval] long *plSenderIdType);
    [propget] HRESULT SenderIdType([in] long lSenderIdType);
    HRESULT Send(
        [in] IMSMQQueue *DestinationQueue,
        [in, optional] VARIANT *Transaction
    );
    HRESULT AttachCurrentSecurityContext();
};

[
    uuid(D9933BE0-A567-11D2-B0F3-00E02C074F6B),
    hidden, dual, odl
]
interface IMSMQMessage2 : IDispatch {
    [propget] HRESULT Class([out, retval] long *plClass);
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT AuthLevel([out, retval] long *plAuthLevel);
    [propput] HRESULT AuthLevel([in] long lAuthLevel);
    [propget] HRESULT IsAuthenticated(
        [out, retval] short *pisAuthenticated
    );
    [propget] HRESULT Delivery([out, retval] long *plDelivery);
    [propput] HRESULT Delivery([in] long lDelivery);
    [propget] HRESULT Trace([out, retval] long *plTrace);
    [propput] HRESULT Trace([in] long lTrace);
    [propget] HRESULT Priority([out, retval] long *plPriority);
    [propput] HRESULT Priority([in] long lPriority);
    [propget] HRESULT Journal([out, retval] long *plJournal);
    [propput] HRESULT Journal([in] long lJournal);
    [propget] HRESULT ResponseQueueInfo_v1(
        [out, retval] IMSMQQueueInfo **ppqinfoResponse
    );
    [propput] HRESULT ResponseQueueInfo_v1(
        [in] IMSMQQueueInfo *pqinfoResponse
    );
    [propget] HRESULT AppSpecific([out, retval] long *plAppSpecific);
    [propput] HRESULT AppSpecific([in] long lAppSpecific);
    [propget] HRESULT SourceMachineGuid(
        [out, retval] BSTR *pbstrGuidSrcMachine
    );
    [propget] HRESULT BodyLength([out, retval] long *pcbBody);
    [propget] HRESULT Body([out, retval] VARIANT *pvarBody);
    [propput] HRESULT Body([in] VARIANT varBody);
    [propget] HRESULT AdminQueueInfo_v1(
        [out, retval] IMSMQQueueInfo **ppqinfoAdmin
    );
    [propput] HRESULT AdminQueueInfo_v1([in] IMSMQQueueInfo *pqinfoAdmin);
    [propget] HRESULT Id([out, retval] VARIANT *pvarMsgId);
    [propget] HRESULT CorrelationId([out, retval] VARIANT *pvarMsgId);
    [propput] HRESULT CorrelationId([in] VARIANT varMsgId);
    [propget] HRESULT Ack([out, retval] long *plAck);
    [propput] HRESULT Ack([in] long lAck);
    [propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
    [propput] HRESULT Label([in] BSTR bstrLabel);
    [propget] HRESULT MaxTimeToReachQueue(
        [out, retval] long *plMaxTimeToReachQueue
    );
};

```

```

[propput] HRESULT MaxTimeToReachQueue([in] long lMaxTimeToReachQueue);
[propget] HRESULT MaxTimeToReceive(
    [out, retval] long *plMaxTimeToReceive
);
[propput] HRESULT MaxTimeToReceive([in] long lMaxTimeToReceive);
[propget] HRESULT HashAlgorithm([out, retval] long *plHashAlg);
[propput] HRESULT HashAlgorithm([in] long lHashAlg);
[propget] HRESULT EncryptAlgorithm([out, retval] long *plEncryptAlg);
[propput] HRESULT EncryptAlgorithm([in] long lEncryptAlg);
[propget] HRESULT SentTime([out, retval] VARIANT *pvarSentTime);
[propget] HRESULT ArrivedTime([out, retval] VARIANT *plArrivedTime);
[propget] HRESULT DestinationQueueInfo(
    [out, retval] IMMQQueueInfo2 **ppqinfoDest
);
[propget] HRESULT SenderCertificate(
    [out, retval] VARIANT *pvarSenderCert
);
[propput] HRESULT SenderCertificate([in] VARIANT varSenderCert);
[propget] HRESULT SenderId([out, retval] VARIANT *pvarSenderId);
[propget] HRESULT SenderIdType([out, retval] long *plSenderIdType);
[propput] HRESULT SenderIdType([in] long lSenderIdType);
HRESULT Send(
    [in] IMMQQueue2 *DestinationQueue,
    [in, optional] VARIANT *Transaction);
HRESULT AttachCurrentSecurityContext();
[propget] HRESULT SenderVersion([out, retval] long *plSenderVersion);
[propget] HRESULT Extension([out, retval] VARIANT *pvarExtension);
[propput] HRESULT Extension([in] VARIANT varExtension);
[propget] HRESULT ConnectorTypeGuid(
    [out, retval] BSTR *pbstrGuidConnectorType
);
[propput] HRESULT ConnectorTypeGuid([in] BSTR bstrGuidConnectorType);
[propget] HRESULT TransactionStatusQueueInfo(
    [out, retval] IMMQQueueInfo2 **ppqinfoXactStatus
);
[propget] HRESULT DestinationSymmetricKey(
    [out, retval] VARIANT *pvarDestSymmKey
);
[propput] HRESULT DestinationSymmetricKey([in] VARIANT varDestSymmKey);
[propget] HRESULT Signature([out, retval] VARIANT *pvarSignature);
[propput] HRESULT Signature([in] VARIANT varSignature);
[propget] HRESULT AuthenticationProviderType(
    [out, retval] long *plAuthProvType
);
[propput] HRESULT AuthenticationProviderType([in] long lAuthProvType);
[propget] HRESULT AuthenticationProviderName(
    [out, retval] BSTR *pbstrAuthProvName
);
[propput] HRESULT AuthenticationProviderName(
    [in] BSTR bstrAuthProvName
);
[propput] HRESULT SenderId([in] VARIANT varSenderId);
[propget] HRESULT MsgClass([out, retval] long *plMsgClass);
[propput] HRESULT MsgClass([in] long lMsgClass);
[propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
[propget] HRESULT TransactionId([out, retval] VARIANT *pvarXactId);
[propget] HRESULT IsFirstInTransaction(
    [out, retval] short *pisFirstInXact
);
[propput] HRESULT IsLastInTransaction([out, retval] short *pisLastInXact);
[propget] HRESULT ResponseQueueInfo(
    [out, retval] IMMQQueueInfo2 **ppqinfoResponse
);
[propputref] HRESULT ResponseQueueInfo(
    [in] IMMQQueueInfo2 *pqinfoResponse
);

```

```

);
[propget] HRESULT AdminQueueInfo(
    [out, retval] IMSMQQueueInfo2 **ppqinfoAdmin
);
[propputref] HRESULT AdminQueueInfo([in] IMSMQQueueInfo2 *pqinfoAdmin);
[propget] HRESULT ReceivedAuthenticationLevel(
    [out, retval] short *psReceivedAuthenticationLevel
);
};

[
    uuid(eba96b1a-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQMessage3 : IDispatch {
    [propget] HRESULT Class([out, retval] long *plClass);
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT AuthLevel([out, retval] long *plAuthLevel);
    [propput] HRESULT AuthLevel([in] long lAuthLevel);
    [propget] HRESULT IsAuthenticated(
        [out, retval] short *pisAuthenticated
    );
    [propget] HRESULT Delivery([out, retval] long *plDelivery);
    [propput] HRESULT Delivery([in] long lDelivery);
    [propget] HRESULT Trace([out, retval] long *plTrace);
    [propput] HRESULT Trace([in] long lTrace);
    [propget] HRESULT Priority([out, retval] long *plPriority);
    [propput] HRESULT Priority([in] long lPriority);
    [propget] HRESULT Journal([out, retval] long *plJournal);
    [propput] HRESULT Journal([in] long lJournal);
    [propget] HRESULT ResponseQueueInfo_v1(
        [out, retval] IMSMQQueueInfo **ppqinfoResponse
    );
    [propput] HRESULT ResponseQueueInfo_v1(
        [in] IMSMQQueueInfo *pqinfoResponse
    );
    [propget] HRESULT AppSpecific([out, retval] long *plAppSpecific);
    [propput] HRESULT AppSpecific([in] long lAppSpecific);
    [propget] HRESULT SourceMachineGuid(
        [out, retval] BSTR *pbstrGuidSrcMachine
    );
    [propget] HRESULT BodyLength([out, retval] long *pcbBody);
    [propget] HRESULT Body([out, retval] VARIANT *pvarBody);
    [propput] HRESULT Body([in] VARIANT varBody);
    [propget] HRESULT AdminQueueInfo_v1(
        [out, retval] IMSMQQueueInfo **ppqinfoAdmin
    );
    [propput] HRESULT AdminQueueInfo_v1(
        [in] IMSMQQueueInfo *pqinfoAdmin
    );
    [propget] HRESULT Id([out, retval] VARIANT *pvarMsgId);
    [propget] HRESULT CorrelationId([out, retval] VARIANT *pvarMsgId);
    [propput] HRESULT CorrelationId([in] VARIANT varMsgId);
    [propget] HRESULT Ack([out, retval] long *plAck);
    [propput] HRESULT Ack([in] long lAck);
    [propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
    [propput] HRESULT Label([in] BSTR bstrLabel);
    [propget] HRESULT MaxTimeToReachQueue(
        [out, retval] long *plMaxTimeToReachQueue
    );
    [propput] HRESULT MaxTimeToReachQueue([in] long lMaxTimeToReachQueue);
    [propget] HRESULT MaxTimeToReceive(
        [out, retval] long *plMaxTimeToReceive
    );
};

```



```

[propput] HRESULT MaxTimeToReceive([in] long lMaxTimeToReceive);
[propget] HRESULT HashAlgorithm([out, retval] long *plHashAlg);
[propput] HRESULT HashAlgorithm([in] long lHashAlg);
[propget] HRESULT EncryptAlgorithm([out, retval] long *plEncryptAlg);
[propput] HRESULT EncryptAlgorithm([in] long lEncryptAlg);
[propget] HRESULT SentTime([out, retval] VARIANT *pvarSentTime);
[propget] HRESULT ArrivedTime([out, retval] VARIANT *plArrivedTime);
[propget] HRESULT DestinationQueueInfo(
    [out, retval] IMMQQueueInfo3 **ppqinfoDest
);
[propget] HRESULT SenderCertificate(
    [out, retval] VARIANT *pvarSenderCert
);
[propput] HRESULT SenderCertificate([in] VARIANT varSenderCert);
[propget] HRESULT SenderId([out, retval] VARIANT *pvarSenderId);
[propget] HRESULT SenderIdType([out, retval] long *plSenderIdType);
[propput] HRESULT SenderIdType([in] long lSenderIdType);
HRESULT Send(
    [in] IDispatch *DestinationQueue,
    [in, optional] VARIANT *Transaction
);
HRESULT AttachCurrentSecurityContext();
[propget] HRESULT SenderVersion([out, retval] long *plSenderVersion);
[propget] HRESULT Extension([out, retval] VARIANT *pvarExtension);
[propput] HRESULT Extension([in] VARIANT varExtension);
[propget] HRESULT ConnectorTypeGuid(
    [out, retval] BSTR *pbstrGuidConnectorType
);
[propput] HRESULT ConnectorTypeGuid([in] BSTR bstrGuidConnectorType);
[propget] HRESULT TransactionStatusQueueInfo(
    [out, retval] IMMQQueueInfo3 **ppqinfoXactStatus
);
[propget] HRESULT DestinationSymmetricKey(
    [out, retval] VARIANT *pvarDestSymmKey
);
[propput] HRESULT DestinationSymmetricKey([in] VARIANT varDestSymmKey);
[propget] HRESULT Signature([out, retval] VARIANT *pvarSignature);
[propput] HRESULT Signature([in] VARIANT varSignature);
[propget] HRESULT AuthenticationProviderType(
    [out, retval] long *plAuthProvType
);
[propput] HRESULT AuthenticationProviderType([in] long lAuthProvType);
[propget] HRESULT AuthenticationProviderName(
    [out, retval] BSTR *pbstrAuthProvName
);
[propput] HRESULT AuthenticationProviderName(
    [in] BSTR bstrAuthProvName
);
[propput] HRESULT SenderId([in] VARIANT varSenderId);
[propget] HRESULT MsgClass([out, retval] long *plMsgClass);
[propput] HRESULT MsgClass([in] long lMsgClass);
[propget] HRESULT Properties(
    [out, retval] IDispatch **ppcolProperties
);
[propget] HRESULT TransactionId([out, retval] VARIANT *pvarXactId);
[propget] HRESULT IsFirstInTransaction(
    [out, retval] short *pisFirstInXact
);
[propget] HRESULT IsLastInTransaction(
    [out, retval] short *pisLastInXact
);
[propget] HRESULT ResponseQueueInfo_v2(
    [out, retval] IMMQQueueInfo2 **ppqinfoResponse
);
[propput] HRESULT ResponseQueueInfo_v2(

```

```

        [in] IMSMQQueueInfo2 *pqinfoResponse
    );
    [propget] HRESULT AdminQueueInfo_v2(
        [out, retval] IMSMQQueueInfo2 **ppqinfoAdmin
    );
    [propput] HRESULT AdminQueueInfo_v2([in] IMSMQQueueInfo2 *pqinfoAdmin);
    [propget] HRESULT ReceivedAuthenticationLevel(
        [out, retval] short *psReceivedAuthenticationLevel
    );
    [propget] HRESULT ResponseQueueInfo(
        [out, retval] IMSMQQueueInfo3 **ppqinfoResponse
    );
    [propputref] HRESULT ResponseQueueInfo(
        [in] IMSMQQueueInfo3 *pqinfoResponse
    );
    [propget] HRESULT AdminQueueInfo(
        [out, retval] IMSMQQueueInfo3 **ppqinfoAdmin
    );
    [propputref] HRESULT AdminQueueInfo([in] IMSMQQueueInfo3 *pqinfoAdmin);
    [propget] HRESULT ResponseDestination(
        [out, retval] IDispatch **ppdestRespons
    );
    [propputref] HRESULT ResponseDestination([in] IDispatch *pdestResponse);
    [propget] HRESULT Destination(
        [out, retval] IDispatch **ppdestDestination
    );
    [propget] HRESULT LookupId([out, retval] VARIANT *pvarLookupId);
    [propget] HRESULT IsAuthenticated2(
        [out, retval] VARIANT_BOOL *pisAuthenticated
    );
    [propget] HRESULT IsFirstInTransaction2(
        [out, retval] VARIANT_BOOL *pisFirstInXact
    );
    [propget] HRESULT IsLastInTransaction2(
        [out, retval] VARIANT_BOOL *pisLastInXact
    );
    HRESULT AttachCurrentSecurityContext2();
    [propget] HRESULT SoapEnvelope([out, retval] BSTR *pbstrSoapEnvelope);
    [propget] HRESULT CompoundMessage(
        [out, retval] VARIANT *pvarCompoundMessage
    );
    [propput] HRESULT SoapHeader([in] BSTR bstrSoapHeader);
    [propput] HRESULT SoapBody([in] BSTR bstrSoapBody);
};

[
    uuid(eba96b23-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQMessage4 : IDispatch {
    [propget] HRESULT Class([out, retval] long *plClass);
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT AuthLevel([out, retval] long *plAuthLevel);
    [propput] HRESULT AuthLevel([in] long lAuthLevel);
    [propget] HRESULT IsAuthenticated(
        [out, retval] short *pisAuthenticated
    );
    [propget] HRESULT Delivery([out, retval] long *plDelivery);
    [propput] HRESULT Delivery([in] long lDelivery);
    [propget] HRESULT Trace([out, retval] long *plTrace);
    [propput] HRESULT Trace([in] long lTrace);
    [propget] HRESULT Priority([out, retval] long *plPriority);
    [propput] HRESULT Priority([in] long lPriority);
    [propget] HRESULT Journal([out, retval] long *plJournal);
};

```

```

[propput] HRESULT Journal([in] long lJournal);
[propget] HRESULT ResponseQueueInfo_v1(
    [out, retval] IMSMQQueueInfo *ppqinfoResponse
);
[propput] HRESULT ResponseQueueInfo_v1(
    [in] IMSMQQueueInfo *pqinfoResponse
);
[propget] HRESULT AppSpecific([out, retval] long *plAppSpecific);
[propput] HRESULT AppSpecific([in] long lAppSpecific);
[propget] HRESULT SourceMachineGuid(
    [out, retval] BSTR *pbstrGuidSrcMachine
);
[propget] HRESULT BodyLength([out, retval] long *pcbBody);
[propget] HRESULT Body([out, retval] VARIANT *pvarBody);
[propput] HRESULT Body([in] VARIANT varBody);
[propget] HRESULT AdminQueueInfo_v1(
    [out, retval] IMSMQQueueInfo *ppqinfoAdmin
);
[propput] HRESULT AdminQueueInfo_v1([in] IMSMQQueueInfo *pqinfoAdmin);
[propget] HRESULT Id([out, retval] VARIANT *pvarMsgId);
[propget] HRESULT CorrelationId([out, retval] VARIANT *pvarMsgId);
[propput] HRESULT CorrelationId([in] VARIANT varMsgId);
[propget] HRESULT Ack([out, retval] long *plAck);
[propput] HRESULT Ack([in] long lAck);
[propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
[propput] HRESULT Label([in] BSTR bstrLabel);
[propget] HRESULT MaxTimeToReachQueue(
    [out, retval] long *plMaxTimeToReachQueue
);
[propput] HRESULT MaxTimeToReachQueue([in] long lMaxTimeToReachQueue);
[propget] HRESULT MaxTimeToReceive(
    [out, retval] long *plMaxTimeToReceive
);
[propput] HRESULT MaxTimeToReceive([in] long lMaxTimeToReceive);
[propget] HRESULT HashAlgorithm([out, retval] long *plHashAlg);
[propput] HRESULT HashAlgorithm([in] long lHashAlg);
[propget] HRESULT EncryptAlgorithm([out, retval] long *plEncryptAlg);
[propput] HRESULT EncryptAlgorithm([in] long lEncryptAlg);
[propget] HRESULT SentTime([out, retval] VARIANT *pvarSentTime);
[propput] HRESULT ArrivedTime([out, retval] VARIANT *plArrivedTime);
[propget] HRESULT DestinationQueueInfo(
    [out, retval] IMSMQQueueInfo4 *ppqinfoDest
);
[propput] HRESULT SenderCertificate(
    [out, retval] VARIANT *pvarSenderCert
);
[propput] HRESULT SenderCertificate([in] VARIANT varSenderCert);
[propget] HRESULT SenderId([out, retval] VARIANT *pvarSenderId);
[propput] HRESULT SenderIdType([out, retval] long *plSenderIdType);
[propput] HRESULT SenderIdType([in] long lSenderIdType);
HRESULT Send(
    [in] IDispatch *DestinationQueue,
    [in, optional] VARIANT *Transaction);
HRESULT AttachCurrentSecurityContext();
[propget] HRESULT SenderVersion([out, retval] long *plSenderVersion);
[propget] HRESULT Extension([out, retval] VARIANT *pvarExtension);
[propput] HRESULT Extension([in] VARIANT varExtension);
[propget] HRESULT ConnectorTypeGuid(
    [out, retval] BSTR *pbstrGuidConnectorType
);
[propput] HRESULT ConnectorTypeGuid([in] BSTR bstrGuidConnectorType);
[propget] HRESULT TransactionStatusQueueInfo(
    [out, retval] IMSMQQueueInfo4 *ppqinfoXactStatus
);
[propput] HRESULT DestinationSymmetricKey(

```

```

        [out, retval] VARIANT *pvarDestSymmKey
    );
    [propput] HRESULT DestinationSymmetricKey([in] VARIANT varDestSymmKey);
    [propget] HRESULT Signature([out, retval] VARIANT *pvarSignature);
    [propput] HRESULT Signature([in] VARIANT varSignature);
    [propget] HRESULT AuthenticationProviderType(
        [out, retval] long *plAuthProvType
    );
    [propput] HRESULT AuthenticationProviderType([in] long lAuthProvType);
    [propget] HRESULT AuthenticationProviderName(
        [out, retval] BSTR *pbstrAuthProvName
    );
    [propput] HRESULT AuthenticationProviderName(
        [in] BSTR bstrAuthProvName
    );
    [propput] HRESULT SenderId([in] VARIANT varSenderId);
    [propget] HRESULT MsgClass([out, retval] long *plMsgClass);
    [propput] HRESULT MsgClass([in] long lMsgClass);
    [propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
    [propget] HRESULT TransactionId([out, retval] VARIANT *pvarXactId);
    [propget] HRESULT IsFirstInTransaction(
        [out, retval] short *pisFirstInXact
    );
    [propget] HRESULT IsLastInTransaction(
        [out, retval] short *pisLastInXact
    );
    [propget] HRESULT ResponseQueueInfo_v2(
        [out, retval] IMSMQQueueInfo2 **ppqinfoResponse
    );
    [propput] HRESULT ResponseQueueInfo v2(
        [in] IMSMQQueueInfo2 *pqinfoResponse
    );
    [propget] HRESULT AdminQueueInfo_v2(
        [out, retval] IMSMQQueueInfo2 **ppqinfoAdmin
    );
    [propput] HRESULT AdminQueueInfo v2(
        [in] IMSMQQueueInfo2 *pqinfoAdmin
    );
    [propget] HRESULT ReceivedAuthenticationLevel(
        [out, retval] short *psReceivedAuthenticationLevel
    );
    [propget] HRESULT ResponseQueueInfo(
        [out, retval] IMSMQQueueInfo4 **ppqinfoResponse
    );
    [propputref] HRESULT ResponseQueueInfo(
        [in] IMSMQQueueInfo4 *pqinfoResponse
    );
    [propget] HRESULT AdminQueueInfo(
        [out, retval] IMSMQQueueInfo4 **ppqinfoAdmin
    );
    [propputref] HRESULT AdminQueueInfo([in] IMSMQQueueInfo4 *pqinfoAdmin);
    [propget] HRESULT ResponseDestination(
        [out, retval] IDispatch **ppdestResponse
    );
    [propputref] HRESULT ResponseDestination([in] IDispatch *pdestResponse);
    [propget] HRESULT Destination(
        [out, retval] IDispatch **ppdestDestination
    );
    [propget] HRESULT LookupId([out, retval] VARIANT *pvarLookupId);
    [propget] HRESULT IsAuthenticated2(
        [out, retval] VARIANT BOOL *pisAuthenticated
    );
    [propget] HRESULT IsFirstInTransaction2(
        [out, retval] VARIANT BOOL *pisFirstInXact
    );
    );

```

```

    [propget] HRESULT IsLastInTransaction2(
        [out, retval] VARIANT_BOOL *pisLastInXact
    );
    HRESULT AttachCurrentSecurityContext2();
    [propget] HRESULT SoapEnvelope([out, retval] BSTR *pbstrSoapEnvelope);
    [propget] HRESULT CompoundMessage(
        [out, retval] VARIANT *pvarCompoundMessage
    );
    [propput] HRESULT SoapHeader([in] BSTR bstrSoapHeader);
    [propput] HRESULT SoapBody([in] BSTR bstrSoapBody);
};
[
    uuid(D7D6E075-DCCD-11d0-AA4B-0060970DEBAE),
]
coclass MSMQMessage {
    interface IMSMQMessage;
    interface IMSMQMessage2;
    interface IMSMQMessage3;
    [default] interface IMSMQMessage4;
};
[
    uuid(D7D6E076-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, odl
]
interface IMSMQQueue : IDispatch {
    [propget] HRESULT Access([out, retval] long *plAccess);
    [propget] HRESULT ShareMode([out, retval] long *plShareMode);
    [propget] HRESULT QueueInfo([out, retval] IMSMQQueueInfo **ppqinfo);
    [propget] HRESULT Handle([out, retval] long *plHandle);
    [propget] HRESULT IsOpen([out, retval] short *pisOpen);
    HRESULT Close();
    HRESULT Receive(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT Peek (
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT EnableNotification(
        [in] IMSMQEvent *Event,
        [in, optional] VARIANT *Cursor,
        [in, optional] VARIANT *ReceiveTimeout
    );
    HRESULT Reset();
    HRESULT ReceiveCurrent(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT PeekNext(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT PeekCurrent(
        [in, optional] VARIANT *WantDestinationQueue,

```

```

        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
};
[
    uuid(EF0574E0-06D8-11D3-B100-00E02C074F6B),
    hidden, dual, odl
]
interface IMSMQQueue2 : IDispatch {
    [propget] HRESULT Access([out, retval] long *plAccess);
    [propget] HRESULT ShareMode([out, retval] long *plShareMode);
    [propget] HRESULT QueueInfo([out, retval] IMSMQQueueInfo2 **ppqinfo);
    [propget] HRESULT Handle([out, retval] long *plHandle);
    [propget] HRESULT IsOpen([out, retval] short *pisOpen);
    HRESULT Close();
    HRESULT Receive v1(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT Peek v1(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT EnableNotification(
        [in] IMSMQEvent2 *Event,
        [in, optional] VARIANT *Cursor,
        [in, optional] VARIANT *ReceiveTimeout
    );
    HRESULT Reset();
    HRESULT ReceiveCurrent v1(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT PeekNext v1(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT PeekCurrent v1(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT Receive(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage2 **ppmsg
    );
    HRESULT Peek(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,

```

```

        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage2 **ppmsg
    );
    HRESULT ReceiveCurrent(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage2 **ppmsg
    );
    HRESULT PeekNext(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage2 **ppmsg
    );
    HRESULT PeekCurrent(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage2 **ppmsg
    );
    [propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
};

[
    uuid(eba96b1b-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQQueue3 : IDispatch {
    [propget] HRESULT Access([out, retval] long *plAccess);
    [propget] HRESULT ShareMode([out, retval] long *plShareMode);
    [propget] HRESULT QueueInfo([out, retval] IMSMQQueueInfo3 **ppqinfo);
    [propget] HRESULT Handle([out, retval] long *plHandle);
    [propget] HRESULT IsOpen([out, retval] short *pisOpen);
    HRESULT Close();
    HRESULT Receive v1(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT Peek v1(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
    HRESULT EnableNotification(
        [in] IMSMQEvent3 *Event,
        [in, optional] VARIANT *Cursor,
        [in, optional] VARIANT *ReceiveTimeout
    );
    HRESULT Reset();
    HRESULT ReceiveCurrent v1(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQMessage **ppmsg
    );
};

```

```

);
HRESULT PeekNext_v1(
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *ReceiveTimeout,
    [out, retval] IMSMQMessage **ppmsg
);
HRESULT PeekCurrent_v1(
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *ReceiveTimeout,
    [out, retval] IMSMQMessage **ppmsg
);
HRESULT Receive(
    [in, optional] VARIANT *Transaction,
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *ReceiveTimeout,
    [in, optional] VARIANT *WantConnectorType,
    [out, retval] IMSMQMessage3 **ppmsg
);
HRESULT Peek(
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *ReceiveTimeout,
    [in, optional] VARIANT *WantConnectorType,
    [out, retval] IMSMQMessage3 **ppmsg
);
HRESULT ReceiveCurrent(
    [in, optional] VARIANT *Transaction,
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *ReceiveTimeout,
    [in, optional] VARIANT *WantConnectorType,
    [out, retval] IMSMQMessage3 **ppmsg
);
HRESULT PeekNext(
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *ReceiveTimeout,
    [in, optional] VARIANT *WantConnectorType,
    [out, retval] IMSMQMessage3 **ppmsg
);
HRESULT PeekCurrent(
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *ReceiveTimeout,
    [in, optional] VARIANT *WantConnectorType,
    [out, retval] IMSMQMessage3 **ppmsg
);
[propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
[propget] HRESULT Handle2([out, retval] VARIANT *pvarHandle);
HRESULT ReceiveByLookupId(
    [in] VARIANT LookupId,
    [in, optional] VARIANT *Transaction,
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,
    [in, optional] VARIANT *WantConnectorType,
    [out, retval] IMSMQMessage3 **ppmsg
);
HRESULT ReceiveNextByLookupId(
    [in] VARIANT LookupId,
    [in, optional] VARIANT *Transaction,
    [in, optional] VARIANT *WantDestinationQueue,
    [in, optional] VARIANT *WantBody,

```



```

        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT ReceivePreviousByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT ReceiveFirstByLookupId(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT ReceiveLastByLookupId(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT PeekByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT PeekNextByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT PeekPreviousByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT PeekFirstByLookupId(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT PeekLastByLookupId(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage3 **ppmsg
    );
    HRESULT Purge();
    [propget] HRESULT IsOpen2([out, retval] VARIANT BOOL *pisOpen);
};

[
    uuid(eba96b20-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl

```

```

]
interface IMSMQueue4 : IDispatch {
    [propget] HRESULT Access([out, retval] long *plAccess);
    [propget] HRESULT ShareMode([out, retval] long *plShareMode);
    [propget] HRESULT QueueInfo([out, retval] IMSMQueueInfo4 **ppqinfo);
    [propget] HRESULT Handle([out, retval] long *plHandle);
    [propget] HRESULT IsOpen([out, retval] short *pisOpen);
    HRESULT Close();
    HRESULT Receive_v1(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQueueMessage **ppmsg
    );
    HRESULT Peek_v1(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQueueMessage **ppmsg
    );
    HRESULT EnableNotification(
        [in] IMSMQueueEvent3 *Event,
        [in, optional] VARIANT *Cursor,
        [in, optional] VARIANT *ReceiveTimeout
    );
    HRESULT Reset();
    HRESULT ReceiveCurrent_v1(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQueueMessage **ppmsg
    );
    HRESULT PeekNext_v1(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQueueMessage **ppmsg
    );
    HRESULT PeekCurrent_v1(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [out, retval] IMSMQueueMessage **ppmsg
    );
    HRESULT Receive(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQueueMessage4 **ppmsg
    );
    HRESULT Peek(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQueueMessage4 **ppmsg
    );
    HRESULT ReceiveCurrent(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,

```

```

        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT PeekNext(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT PeekCurrent(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *ReceiveTimeout,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
    [propget] HRESULT Handle2(
        [out, retval] VARIANT *pvarHandle
    );
    HRESULT ReceiveByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT ReceiveNextByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT ReceivePreviousByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT ReceiveFirstByLookupId(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT ReceiveLastByLookupId(
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT PeekByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *WantDestinationQueue,

```

```

        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT PeekNextByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT PeekPreviousByLookupId(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT PeekFirstByLookupId(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT PeekLastByLookupId(
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
    HRESULT Purge();
    [propget] HRESULT IsOpen2([out, retval] VARIANT_BOOL *pisOpen);
    HRESULT ReceiveByLookupIdAllowPeek(
        [in] VARIANT LookupId,
        [in, optional] VARIANT *Transaction,
        [in, optional] VARIANT *WantDestinationQueue,
        [in, optional] VARIANT *WantBody,
        [in, optional] VARIANT *WantConnectorType,
        [out, retval] IMSMQMessage4 **ppmsg
    );
};

[
    uuid(D7D6E079-DCCD-11d0-AA4B-0060970DEBAE),
]
coclass MSMQQueue {
    interface IMSMQQueue;
    interface IMSMQQueue2;
    interface IMSMQQueue3;
    [default] interface IMSMQQueue4;
};

[
    uuid(D7AB3341-C9D3-11d1-BB47-0080C7C5A2C0),
    hidden, dual, odl
]
interface IMSMQPrivateEvent : IDispatch {
    [propget] HRESULT Hwnd([out, retval] long *phwnd);
    HRESULT FireArrivedEvent(
        [in] IMSMQQueue *pq,
        [in] long msgcursor
    );
    HRESULT FireArrivedErrorEvent(
        [in] IMSMQQueue *pq,
        [in] HRESULT hrStatus,
        [in] long msgcursor
    );
};

```

```

    );
}

[
    uuid(D7D6E077-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, odl
]
interface IMSMQEvent : IDispatch {
}

[
    uuid(eba96b12-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQEvent2 : IMSMQEvent {
    [propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
}

[
    uuid(eba96b1c-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQEvent3 : IMSMQEvent2 {
}

[
    uuid(D7D6E07A-DCCD-11d0-AA4B-0060970DEBAE)
]
coclass MSMQEvent {
    interface IMSMQEvent;
    interface IMSMQEvent2;
    [default] interface IMSMQEvent3;
    interface IMSMQPrivateEvent;
};

[
    uuid(D7D6E07B-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, odl
]
interface IMSMQQueueInfo : IDispatch {
    [propget] HRESULT QueueGuid([out, retval] BSTR *pbstrGuidQueue);
    [propget] HRESULT ServiceTypeGuid(
        [out, retval] BSTR *pbstrGuidServiceType
    );
    [propput] HRESULT ServiceTypeGuid([in] BSTR bstrGuidServiceType);
    [propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
    [propput] HRESULT Label([in] BSTR bstrLabel);
    [propget] HRESULT PathName([out, retval] BSTR *pbstrPathName);
    [propput] HRESULT PathName([in] BSTR bstrPathName);
    [propget] HRESULT FormatName([out, retval] BSTR *pbstrFormatName);
    [propput] HRESULT FormatName([in] BSTR bstrFormatName);
    [propget] HRESULT IsTransactional(
        [out, retval] short *pisTransactional
    );
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT Journal([out, retval] long *plJournal);
    [propput] HRESULT Journal([in] long lJournal);
    [propget] HRESULT Quota([out, retval] long *plQuota);
    [propput] HRESULT Quota([in] long lQuota);
    [propget] HRESULT BasePriority([out, retval] long *plBasePriority);
    [propput] HRESULT BasePriority([in] long lBasePriority);
    [propget] HRESULT CreateTime([out, retval] VARIANT *pvarCreateTime);
    [propget] HRESULT ModifyTime([out, retval] VARIANT *pvarModifyTime);
    [propget] HRESULT Authenticate([out, retval] long *plAuthenticate);
}

```

```

    [propput] HRESULT Authenticate([in] long lAuthenticate);
    [propget] HRESULT JournalQuota([out, retval] long *plJournalQuota);
    [propput] HRESULT JournalQuota([in] long lJournalQuota);
    [propget] HRESULT IsWorldReadable(
        [out, retval] short *pisWorldReadable
    );
    HRESULT Create(
        [in, optional] VARIANT *IsTransactional,
        [in, optional] VARIANT *IsWorldReadable
    );
    HRESULT Delete();
    HRESULT Open(
        [in] long Access,
        [in] long ShareMode,
        [out, retval] IMSMQQueue **ppq
    );
    HRESULT Refresh();
    HRESULT Update();
};
[
    uuid(FD174A80-89CF-11D2-B0F2-00E02C074F6B),
    hidden, dual, odl
]
interface IMSMQQueueInfo2 : IDispatch {
    [propget] HRESULT QueueGuid([out, retval] BSTR *pbstrGuidQueue);
    [propget] HRESULT ServiceTypeGuid(
        [out, retval] BSTR *pbstrGuidServiceType
    );
    [propput] HRESULT ServiceTypeGuid([in] BSTR bstrGuidServiceType);
    [propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
    [propput] HRESULT Label([in] BSTR bstrLabel);
    [propget] HRESULT PathName([out, retval] BSTR *pbstrPathName);
    [propput] HRESULT PathName([in] BSTR bstrPathName);
    [propget] HRESULT FormatName([out, retval] BSTR *pbstrFormatName);
    [propput] HRESULT FormatName([in] BSTR bstrFormatName);
    [propget] HRESULT IsTransactional(
        [out, retval] short *pisTransactional
    );
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT Journal([out, retval] long *plJournal);
    [propput] HRESULT Journal([in] long lJournal);
    [propget] HRESULT Quota([out, retval] long *plQuota);
    [propput] HRESULT Quota([in] long lQuota);
    [propget] HRESULT BasePriority([out, retval] long *plBasePriority);
    [propput] HRESULT BasePriority([in] long lBasePriority);
    [propget] HRESULT CreateTime([out, retval] VARIANT *pvarCreateTime);
    [propget] HRESULT ModifyTime([out, retval] VARIANT *pvarModifyTime);
    [propget] HRESULT Authenticate([out, retval] long *plAuthenticate);
    [propput] HRESULT Authenticate([in] long lAuthenticate);
    [propget] HRESULT JournalQuota([out, retval] long *plJournalQuota);
    [propput] HRESULT JournalQuota([in] long lJournalQuota);
    [propget] HRESULT IsWorldReadable(
        [out, retval] short *pisWorldReadable
    );
    HRESULT Create(
        [in, optional] VARIANT *IsTransactional,
        [in, optional] VARIANT *IsWorldReadable
    );
    HRESULT Delete();
    HRESULT Open(
        [in] long Access,
        [in] long ShareMode,
        [out, retval] IMSMQQueue2 **ppq
    );
};

```

```

HRESULT Refresh();
HRESULT Update();
[propget] HRESULT PathNameDNS([out, retval] BSTR *pbstrPathNameDNS);
[propget] HRESULT Properties(
    [out, retval] IDispatch **ppcolProperties
);
[propget] HRESULT Security([out, retval] VARIANT *pvarSecurity);
[propget] HRESULT Security([in] VARIANT varSecurity);
};

[
    uuid(eba96b1d-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQQueueInfo3 : IDispatch {
    [propget] HRESULT QueueGuid([out, retval] BSTR *pbstrGuidQueue);
    [propget] HRESULT ServiceTypeGuid(
        [out, retval] BSTR *pbstrGuidServiceType
    );
    [propput] HRESULT ServiceTypeGuid([in] BSTR bstrGuidServiceType);
    [propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
    [propput] HRESULT Label([in] BSTR bstrLabel);
    [propget] HRESULT PathName([out, retval] BSTR *pbstrPathName);
    [propput] HRESULT PathName([in] BSTR bstrPathName);
    [propget] HRESULT FormatName([out, retval] BSTR *pbstrFormatName);
    [propput] HRESULT FormatName([in] BSTR bstrFormatName);
    [propget] HRESULT IsTransactional(
        [out, retval] short *pisTransactional
    );
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT Journal([out, retval] long *plJournal);
    [propput] HRESULT Journal([in] long lJournal);
    [propget] HRESULT Quota([out, retval] long *plQuota);
    [propput] HRESULT Quota([in] long lQuota);
    [propget] HRESULT BasePriority([out, retval] long *plBasePriority);
    [propput] HRESULT BasePriority([in] long lBasePriority);
    [propget] HRESULT CreateTime([out, retval] VARIANT *pvarCreateTime);
    [propget] HRESULT ModifyTime([out, retval] VARIANT *pvarModifyTime);
    [propget] HRESULT Authenticate([out, retval] long *plAuthenticate);
    [propput] HRESULT Authenticate([in] long lAuthenticate);
    [propget] HRESULT JournalQuota([out, retval] long *plJournalQuota);
    [propput] HRESULT JournalQuota([in] long lJournalQuota);
    [propget] HRESULT IsWorldReadable(
        [out, retval] short *pisWorldReadable
    );
    HRESULT Create(
        [in, optional] VARIANT *IsTransactional,
        [in, optional] VARIANT *IsWorldReadable
    );
    HRESULT Delete();
    HRESULT Open(
        [in] long Access,
        [in] long ShareMode,
        [out, retval] IMSMQQueue3 **ppq
    );
    HRESULT Refresh();
    HRESULT Update();
    [propget] HRESULT PathNameDNS([out, retval] BSTR *pbstrPathNameDNS);
    [propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
    [propget] HRESULT Security([out, retval] VARIANT *pvarSecurity);
    [hidden, propput] HRESULT Security([in] VARIANT varSecurity);
    [propget] HRESULT IsTransactional2(
        [out, retval] VARIANT BOOL *pisTransactional
    );
};

```

```

    [propget] HRESULT IsWorldReadable2(
        [out, retval] VARIANT_BOOL *pisWorldReadable
    );
    [propget] HRESULT MulticastAddress(
        [out, retval] BSTR *pbstrMulticastAddress
    );
    [propput] HRESULT MulticastAddress([in] BSTR bstrMulticastAddress);
    [propget] HRESULT ADsPath([out, retval] BSTR *pbstrADsPath);
};
[
    uuid(eba96b21-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQQueueInfo4 : IDispatch {
    [propget] HRESULT QueueGuid([out, retval] BSTR *pbstrGuidQueue);
    [propget] HRESULT ServiceTypeGuid(
        [out, retval] BSTR *pbstrGuidServiceType
    );
    [propput] HRESULT ServiceTypeGuid([in] BSTR bstrGuidServiceType);
    [propget] HRESULT Label([out, retval] BSTR *pbstrLabel);
    [propput] HRESULT Label([in] BSTR bstrLabel);
    [propget] HRESULT PathName([out, retval] BSTR *pbstrPathName);
    [propput] HRESULT PathName([in] BSTR bstrPathName);
    [propget] HRESULT FormatName([out, retval] BSTR *pbstrFormatName);
    [propput] HRESULT FormatName([in] BSTR bstrFormatName);
    [propget] HRESULT IsTransactional(
        [out, retval] short *pisTransactional
    );
    [propget] HRESULT PrivLevel([out, retval] long *plPrivLevel);
    [propput] HRESULT PrivLevel([in] long lPrivLevel);
    [propget] HRESULT Journal([out, retval] long *plJournal);
    [propput] HRESULT Journal([in] long lJournal);
    [propget] HRESULT Quota([out, retval] long *plQuota);
    [propput] HRESULT Quota([in] long lQuota);
    [propget] HRESULT BasePriority([out, retval] long *plBasePriority);
    [propput] HRESULT BasePriority([in] long lBasePriority);
    [propget] HRESULT CreateTime([out, retval] VARIANT *pvarCreateTime);
    [propget] HRESULT ModifyTime([out, retval] VARIANT *pvarModifyTime);
    [propget] HRESULT Authenticate([out, retval] long *plAuthenticate);
    [propput] HRESULT Authenticate([in] long lAuthenticate);
    [propget] HRESULT JournalQuota([out, retval] long *plJournalQuota);
    [propput] HRESULT JournalQuota([in] long lJournalQuota);
    [propget] HRESULT IsWorldReadable(
        [out, retval] short *pisWorldReadable
    );
    HRESULT Create(
        [in, optional] VARIANT *IsTransactional,
        [in, optional] VARIANT *IsWorldReadable
    );
    HRESULT Delete();
    HRESULT Open(
        [in] long Access,
        [in] long ShareMode,
        [out, retval] IMSMQQueue4 **ppq
    );
    HRESULT Refresh();
    HRESULT Update();
    [propget] HRESULT PathNameDNS([out, retval] BSTR *pbstrPathNameDNS);
    [propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
    [propget] HRESULT Security([out, retval] VARIANT *pvarSecurity);
    [propput] HRESULT Security([in] VARIANT varSecurity);
    [propget] HRESULT IsTransactional2(
        [out, retval] VARIANT_BOOL *pisTransactional
    );
    [propget] HRESULT IsWorldReadable2(

```



```

        [out, retval] VARIANT_BOOL *pisWorldReadable
    );
    [propget] HRESULT MulticastAddress(
        [out, retval] BSTR *pbstrMulticastAddress
    );
    [propput] HRESULT MulticastAddress([in] BSTR bstrMulticastAddress);
    [propget] HRESULT ADsPath([out, retval] BSTR *pbstrADsPath);
};
[
    uuid(D7D6E07C-DCCD-11d0-AA4B-0060970DEBAE),
]
coclass MSMQueueInfo {
    interface IMSMQueueInfo;
    interface IMSMQueueInfo2;
    interface IMSMQueueInfo3;
    [default] interface IMSMQueueInfo4;
};

[
    uuid(D7D6E07D-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, odl
]
interface IMSMQueueInfos : IDispatch {
    HRESULT Reset();
    HRESULT Next([out, retval] IMSMQueueInfo **ppqinfoNext);
};
[
    uuid(eba96b0f-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQueueInfos2 : IDispatch {
    HRESULT Reset();
    HRESULT Next([out, retval] IMSMQueueInfo2 **ppqinfoNext);
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};
[
    uuid(eba96b1e-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQueueInfos3 : IDispatch {
    HRESULT Reset();
    HRESULT Next([out, retval] IMSMQueueInfo3 **ppqinfoNext);
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};
[
    uuid(eba96b22-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQueueInfos4 : IDispatch {
    HRESULT Reset();
    HRESULT Next([out, retval] IMSMQueueInfo4 **ppqinfoNext);
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};
[
    uuid(D7D6E07E-DCCD-11d0-AA4B-0060970DEBAE),
]
coclass MSMQueueInfos {
    interface IMSMQueueInfos;
    interface IMSMQueueInfos2;
};

```

```

        interface IMSMQQueueInfos3;
        [default] interface IMSMQQueueInfos4;
    };
    [
        uuid(D7D6E07F-DCCD-11d0-AA4B-0060970DEBAE),
        hidden, dual, odl
    ]
    interface IMSMQTransaction : IDispatch {
        [propget] HRESULT Transaction([out, retval] long *plTransaction);
        HRESULT Commit(
            [in, optional] VARIANT *fRetaining,
            [in, optional] VARIANT *grfTC,
            [in, optional] VARIANT *grfRM
        );
        HRESULT Abort(
            [in, optional] VARIANT *fRetaining,
            [in, optional] VARIANT *fAsync
        );
    };
    [
        uuid(2CE0C5B0-6E67-11D2-B0E6-00E02C074F6B),
        hidden, dual, odl
    ]
    interface IMSMQTransaction2 : IMSMQTransaction {
        HRESULT InitNew([in] VARIANT varTransaction);
        [propget] HRESULT Properties(
            [out, retval] IDispatch **ppcolProperties
        );
    };
    [
        uuid(eba96b13-2168-11d3-898c-00e02c074f6b),
        hidden, dual, odl
    ]
    interface IMSMQTransaction3 : IMSMQTransaction2 {
        [propget] HRESULT ITransaction([out, retval] VARIANT *pvarITransaction);
    };
    [
        uuid(D7D6E080-DCCD-11d0-AA4B-0060970DEBAE),
    ]
    coclass MSMQTransaction {
        interface IMSMQTransaction;
        interface IMSMQTransaction2;
        [default] interface IMSMQTransaction3;
    };
    [
        uuid(D7D6E081-DCCD-11d0-AA4B-0060970DEBAE),
        hidden, dual, odl
    ]
    interface IMSMQCoordinatedTransactionDispenser : IDispatch {
        HRESULT BeginTransaction(
            [out, retval] IMSMQTransaction **ptransaction
        );
    };
    [
        uuid(eba96b10-2168-11d3-898c-00e02c074f6b),
        hidden, dual, odl
    ]
    interface IMSMQCoordinatedTransactionDispenser2 : IDispatch {
        HRESULT BeginTransaction(
            [out, retval] IMSMQTransaction2 **ptransaction
        );
        [propget] HRESULT Properties(
            [out, retval] IDispatch **ppcolProperties
        );
    };
};

```

```

[
    uuid(eba96b14-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQCoordinatedTransactionDispenser3 : IDispatch {
    HRESULT BeginTransaction(
        [out, retval] IMSMQTransaction3 **ptransaction
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};

[
    uuid(D7D6E082-DCCD-11d0-AA4B-0060970DEBAE)
]
coclass MSMQCoordinatedTransactionDispenser {
    interface IMSMQCoordinatedTransactionDispenser;
    interface IMSMQCoordinatedTransactionDispenser2;
    [default] interface IMSMQCoordinatedTransactionDispenser3;
};

[
    uuid(D7D6E083-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, odl
]
interface IMSMQTransactionDispenser : IDispatch {
    HRESULT BeginTransaction(
        [out, retval] IMSMQTransaction **ptransaction
    );
};

[
    uuid(eba96b11-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQTransactionDispenser2 : IDispatch {
    HRESULT BeginTransaction(
        [out, retval] IMSMQTransaction2 **ptransaction
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};

[
    uuid(eba96b15-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQTransactionDispenser3 : IDispatch {
    HRESULT BeginTransaction(
        [out, retval] IMSMQTransaction3 **ptransaction
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};

[
    uuid(D7D6E084-DCCD-11d0-AA4B-0060970DEBAE),
]
coclass MSMQTransactionDispenser {
    interface IMSMQTransactionDispenser;
    interface IMSMQTransactionDispenser2;
    [default] interface IMSMQTransactionDispenser3;
};

```

```

};

[
    uuid(D7D6E085-DCCD-11d0-AA4B-0060970DEBAE),
    hidden, dual, odl
]
interface IMSMQApplication : IDispatch {
    HRESULT MachineIdOfMachineName(
        [in] BSTR MachineName,
        [out, retval] BSTR *pbstrGuid
    );
};

[
    uuid(12A30900-7300-11D2-B0E6-00E02C074F6B),
    hidden, dual, odl
]
interface IMSMQApplication2 : IMSMQApplication {
    HRESULT RegisterCertificate(
        [in, optional] VARIANT * Flags,
        [in, optional] VARIANT * ExternalCertificate
    );
    HRESULT MachineNameOfMachineId(
        [in] BSTR bstrGuid,
        [out, retval] BSTR *pbstrMachineName
    );
    [propget] HRESULT MSMQVersionMajor(
        [out, retval] short *psMSMQVersionMajor
    );
    [propget] HRESULT MSMQVersionMinor(
        [out, retval] short *psMSMQVersionMinor
    );
    [propget] HRESULT MSMQVersionBuild(
        [out, retval] short *psMSMQVersionBuild
    );
    [propget] HRESULT IsDsEnabled(
        [out, retval] VARIANT BOOL *pfIsDsEnabled
    );
    [propget] HRESULT Properties(
        [out, retval] IDispatch **ppcolProperties
    );
};

[
    uuid(eba96b1f-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQApplication3 : IMSMQApplication2 {
    HRESULT ActiveQueues([out,retval] VARIANT* pvActiveQueues);
    [propget] HRESULT PrivateQueues([out,retval] VARIANT* pvPrivateQueues);
    [propget] HRESULT DirectoryServiceServer(
        [out,retval] BSTR* pbstrDirectoryServiceServer
    );
    [propget] HRESULT IsConnected(
        [out,retval] VARIANT BOOL* pfIsConnected
    );
    [propget] HRESULT BytesInAllQueues(
        [out,retval] VARIANT* pvBytesInAllQueues
    );
    [propput] HRESULT Machine([in] BSTR bstrMachine);
    [propget] HRESULT Machine([out, retval] BSTR *pbstrMachine);
    HRESULT Connect();
    HRESULT Disconnect();
    HRESULT Tidy();
};

```

```

[
    uuid(D7D6E086-DCCD-11d0-AA4B-0060970DEBAE),
    appobject,
]
coclass MSMQApplication {
    interface IMSMQApplication;
    interface IMSMQApplication2;
    [default] interface IMSMQApplication3;
};

[
    uuid(eba96b16-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQDestination : IDispatch {
    HRESULT Open();
    HRESULT Close();
    [propget] HRESULT IsOpen([out, retval] VARIANT BOOL *pfIsOpen);
    [propget] HRESULT IADs([out, retval] IDispatch **ppIADs);
    [propputref] HRESULT IADs([in] IDispatch *pIADs);
    [propget] HRESULT ADsPath([out, retval] BSTR *pbstrADsPath);
    [propput] HRESULT ADsPath([in] BSTR bstrADsPath);
    [propget] HRESULT PathName([out, retval] BSTR *pbstrPathName);
    [propput] HRESULT PathName([in] BSTR bstrPathName);
    [propget] HRESULT FormatName([out, retval] BSTR *pbstrFormatName);
    [propput] HRESULT FormatName([in] BSTR bstrFormatName);
    [propget] HRESULT Destinations([out, retval] IDispatch **ppDestinations);
    [propputref] HRESULT Destinations([in] IDispatch *pDestinations);
    [propget] HRESULT Properties([out, retval] IDispatch **ppcolProperties);
};

[
    uuid(eba96b17-2168-11d3-898c-00e02c074f6b),
    hidden, dual, odl
]
interface IMSMQPrivateDestination : IDispatch {
    [propget] HRESULT Handle([out, retval] VARIANT *pvarHandle);
    [propput] HRESULT Handle([in] VARIANT varHandle);
};

[
    uuid(eba96b18-2168-11d3-898c-00e02c074f6b),
]
coclass MSMQDestination {
    [default] interface IMSMQDestination;
    interface IMSMQPrivateDestination;
};

[
    uuid(0188AC2F-ECB3-4173-9779-635CA2039C72),
    dual, oleautomation
]
interface IMSMQCollection : IDispatch {
    HRESULT Item(
        [in] VARIANT* Index,
        [out, retval] VARIANT* pvarRet
    );
    [propget] HRESULT Count([out, retval] long* pCount);
    [restricted] HRESULT NewEnum([out, retval] IUnknown** ppunk);
};

[
    uuid(f72b9031-2f0c-43e8-924e-e6052cdc493f),
]
coclass MSMQCollection {

```

```

[default]      interface IMSMQCollection;
};

[
    uuid(BE5F0241-E489-4957-8CC4-A452FCF3E23E),
    hidden, dual
]
interface IMSMQManagement : IDispatch {
    HRESULT Init(
        [in, optional] VARIANT *Machine,
        [in, optional] VARIANT *Pathname,
        [in, optional] VARIANT *FormatName
    );
    [propget] HRESULT FormatName([out, retval] BSTR *pbstrFormatName);
    [propget] HRESULT Machine([out, retval] BSTR *pbstrMachine);
    [propget] HRESULT MessageCount([out, retval] long* plMessageCount);
    [propget] HRESULT ForeignStatus([out,retval] long* plForeignStatus);
    [propget] HRESULT QueueType([out,retval] long* plQueueType);
    [propget] HRESULT IsLocal([out,retval] VARIANT BOOL* pfIsLocal);
    [propget] HRESULT TransactionalStatus(
        [out,retval] long* plTransactionalStatus
    );
    [propget] HRESULT BytesInQueue([out,retval] VARIANT* pvBytesInQueue);
};

[
    uuid(39CE96FE-F4C5-4484-A143-4C2D5D324229),
]
coclass MSMQManagement {
    [default] interface IMSMQManagement;
};

[
    uuid(64C478FB-F9B0-4695-8A7F-439AC94326D3),
    dual
]
interface IMSMQOutgoingQueueManagement : IMSMQManagement {
    [propget] HRESULT State([out,retval] long* plState);
    [propget] HRESULT NextHops([out,retval] VARIANT* pvNextHops);
    HRESULT EodGetSendInfo(
        [out,retval] IMSMQCollection** ppCollection
    );
    HRESULT Resume();
    HRESULT Pause();
    HRESULT EodResend();
};

[
    uuid(0188401c-247a-4fed-99c6-bf14119d7055),
]
coclass MSMQOutgoingQueueManagement {
    [default] interface IMSMQOutgoingQueueManagement;
};

[
    uuid(7FBE7759-5760-444d-B8A5-5E7AB9A84CCE),
    dual
]
interface IMSMQQueueManagement : IMSMQManagement {
    [propget] HRESULT JournalMessageCount(
        [out,retval] long* plJournalMessageCount
    );
    [propget] HRESULT BytesInJournal(
        [out,retval] VARIANT* pvBytesInJournal
    );
};

```

```
        HRESULT EodGetReceiveInfo(  
            [out,retval] VARIANT* pvCollection  
        );  
};  
[uuid(33b6d07e-f27d-42fa-b2d7-bf82e11e9374),  
]  
coclass MSMQQueueManagement {  
    [default]interface IMSMQQueueManagement;  
};
```

7 Appendix B: Windows Behavior

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

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

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

[<1> Section 2.2.2.9:](#) Windows servers permit a maximum of 30 routing hops.

[<2> Section 2.2.2.9:](#) Windows servers generate a MQMSG_CLASS_NACK_SOURCE_COMPUTER_GUID_CHANGED value for each message in an outgoing queue where the server has joined or unjoined a Windows domain prior to the message being successfully delivered.

[<3> Section 2.2.2.13:](#) Windows servers send MQMSG_CLASS_REPORT messages to a [Public Queue] owned by the [Local Queue Manager], where [Label] equals "MQReport Queue" and [Identifier GUID] equals "{55ee8f32-cce9-11cf-b108-0020afd61ce9}". If no such [Public Queue] exists, the report message is not generated.

[<4> Section 2.2.2.16:](#) Windows servers interpret MQMSG_AUTH_LEVEL_ALWAYS differently depending on the version of the server. If the server is Windows Vista, Windows Server 2003, or Windows XP, the value is interpreted as equal to MQMSG_AUTH_LEVEL_SIG30. If the server is Windows 2000, the value is interpreted as equal to MQMSG_AUTH_LEVEL_SIG20. If the server is Windows NT, the value is interpreted as equal to MQMSG_AUTH_LEVEL_SIG10.

[<5> Section 2.2.2.18:](#) This value is only supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003. Windows Vista servers MAY enable this value with a configuration option.

[<6> Section 2.2.2.18:](#) This value is only supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003. Windows Vista servers MAY enable this value with a configuration option.

[<7> Section 2.2.2.18:](#) This value is only supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003. Windows Vista servers MAY enable this value with a configuration option.

[<8> Section 2.2.2.18:](#) This value is only supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003. Windows Vista servers MAY enable this value with a configuration option.

[<9> Section 3.1.1.1:](#) Windows servers use domain membership, and an optional installed component, to enable directory affiliation for a queue manager.

[<10> Section 3.1.1.6:](#) Windows servers register the user certificate automatically during the logon sequence.

<11> [Section 3.1.1.11:](#) Windows servers use a Lightweight Directory Access Protocol (LDAP) URL, as defined in [\[RFC4516\]](#), for the [Directory Queue].[Path] property. For more information, see [\[LDAP\]](#).

<12> [Section 3.1.1.16:](#) Multicast messaging for Windows servers is implemented by the Pragmatic General Multicast (PGM) protocol, as described in [\[MC-MQSRM\]](#) section 1.

<13> [Section 3.1.1.23:](#) The subset of supported values defined by the Message Queuing Binary Protocol Specification are contained in [\[MS-MQOB\]](#) section 2.2.2.4.

<14> [Section 3.2.4.1.1:](#) The Windows API documentation for MSMQApplication.MachineIdOfMachineName includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_ACCESS_DENIED	0xC00E0025
MQ_ERROR_MACHINE_NOT_FOUND	0xC00E000D
MQ_ERROR_NO_DS	0xC00E0013

<15> [Section 3.2.4.2.1:](#) The Windows API documentation for MSMQApplication.RegisterCertificate includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_CANNOT_CREATE_CERT_STORE	0xC00E006F
MQ_ERROR_CANNOT_OPEN_CERT_STORE	0xC00E0070
MQ_ERROR_ILLEGAL_USER	0xC00E0011
MQ_ERROR_INTERNAL_USER_CERT_EXIST	0xC00E002E
MQ_INFORMATION_INTERNAL_USER_CERT_EXIST	0x400E000A

<16> [Section 3.2.4.2.1:](#) Windows clients obtain the reference to user SID by calling the Windows SDK function GetTokenInformation.

<17> [Section 3.2.4.2.2:](#) The Windows API documentation for MSMQApplication.MachineNameOfMachineId includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_ACCESS_DENIED	0xC00E0025
MQ_ERROR_ILLEGAL_PROPERTY_VALUE	0xC00E00018
MQ_ERROR_MACHINE_NOT_FOUND	0xC00E000D
MQ_ERROR_NO_DS	0xC00E0013

[<18> Section 3.2.4.2.3:](#) The major version number for the different MSMQ versions are returned as follows.

MSMQ version	Major version number
2.0	2
3.0	3
4.0	4

[<19> Section 3.2.4.2.4:](#) The minor version number for the different MSMQ versions are returned as follows.

MSMQ version	Minor version number
2.0	0
3.0	0
4.0	1

[<20> Section 3.2.4.2.5:](#) The build version number for the different MSMQ versions are returned as follows. Note that the build number is affected by several factors, including Service Packs, security patches, and customer support patches. The values provided below indicate the build number returned by the most recent widely-distributed build of MSMQ for each version, but other values may be encountered depending on the system configuration.

MSMQ version	Build version number
2.0	802
3.0	1738
4.0	6000

[<21> Section 3.2.4.3.3:](#) Windows clients retrieve the name of the domain controller for the directory service server by calling the Windows SDK function DsGetDcName.

[<22> Section 3.2.4.3.8:](#) The Windows API documentation for MSMQApplication.Connect includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_ACCESS_DENIED	0xC00E0025
MQ_ERROR_SERVICE_NOT_AVAILABLE	0xC00E000B

[<23> Section 3.2.4.3.9:](#) The Windows API documentation for MSMQApplication.Disconnect includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_ACCESS_DENIED	0xC00E0025
MQ_ERROR_SERVICE_NOT_AVAILABL	0xC00E000B

<24> [Section 3.2.4.3.10:](#) The Windows API documentation for MSMQApplication.Tidy includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_ACCESS_DENIED	0xC00E0025
MQ_ERROR_SERVICE_NOT_AVAILABL	0xC00E000B

<25> [Section 3.5.4.1.4:](#) The Windows API documentation for MSMQOutgoingQueueManagement.Resume includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_QUEUE_NOT_ACTIVE	0xC00E0004

<26> [Section 3.5.4.1.5:](#) The Windows API documentation for MSMQOutgoingQueueManagement.Pause includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_QUEUE_NOT_ACTIVE	0xC00E0004

<27> [Section 3.5.4.1.6:](#) The Windows API documentation for MSMQOutgoingQueueManagement.EodResend includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_QUEUE_NOT_ACTIVE	0xC00E0004

<28> [Section 3.6.4.1:](#) The ptransaction parameter type is IMSMQTransaction2 or IMSMQTransaction in the IMSMQTransaction2 or IMSMQTransaction interfaces respectively.

<29> [Section 3.6.4.1:](#) This method is not available in the IMSMQTransactionDispenser interface.

<30> [Section 3.6.4.1.1:](#) The Windows API documentation for MSMQTransactionDispenser.BeginTransaction includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_TRANSACTION_USAGE	0xC00E0050

<31> [Section 3.7.4.1:](#) The ptransaction parameter type is IMSMQTransaction2 or IMSMQTransaction in the IMSMQCoordinatedTransactionDispenser2 or IMSMQCoordinatedTransactionDispenser interfaces respectively.

<32> [Section 3.7.4.1:](#) This method is not available in the IMSMQCoordinatedTransactionDispenser interface.

<33> [Section 3.7.4.1.1:](#) The Windows API documentation for MSMQCoordinatedTransactionDispenser.BeginTransaction includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_TRANSACTION_USAGE	0xC00E0050

<34> [Section 3.7.4.1.1:](#) Windows clients obtain the reference to the MS DTC transaction manager by calling the Windows SDK function DtcGetTransactionManager.

<35> [Section 3.7.4.1.1:](#) Windows clients obtain the reference to the [ITransaction \(section 3.8.4.1\)](#) object by calling the Windows SDK function BeginTransaction on the ITransactionDispenser object, which is obtained by calling DtcGetTransactionManager.

<36> [Section 3.8.4.1.1:](#) The Windows API documentation for ITransaction.Commit includes the following return codes. For descriptions of the following return code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
E_POINTER	0x80000005
E_NOTIMPL	0x80000001

<37> [Section 3.8.4.1.2:](#) The Windows API documentation for ITransaction.Abort includes the following return codes. For descriptions of the following return code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
XACT_S_ASYNC	0x0004D000
XACT_E_NOTTRANSACTION	0x8004D00E
XACT_E_ALREADYINPROGRESS	0x8004D018
XACT_E_CANTRETAIN	0x8004D001

Name	Value
E_FAIL	0x80004005
XACT_E_INDOUBT	0x8004D016
E_UNEXPECTED	0x8000FFFF
XACT_S_ABORTING	0x0004D008
XACT_E_CONNECTION_DOWN	0x8004D01C

<38> [Section 3.8.4.1.3:](#) The Windows API documentation for `ITransaction.GetTransactionInfo` includes the following return codes. For descriptions of the following return code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
XACT_E_NOTTRANSACTION	0x8004D00E
E_UNEXPECTED	0x8000FFFF
E_INVALIDARG	0x80070057

<39> [Section 3.9.4.1.1:](#) The current implementation of `Transaction` returns the pointer to the `Transaction` object in the long format.

<40> [Section 3.10.4.1:](#) `IMSMQQueue3`, `IMSMQQueue2` and `IMSMQQueue` pointers are returned for the `IMSMQQueueInfo3`, `IMSMQQueueInfo2` and `IMSMQQueueInfo` versions of this interface respectively.

<41> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` interface.

<42> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` interface.

<43> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` interface.

<44> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` interface.

<45> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` or `IMSMQQueueInfo2` interfaces.

<46> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` or `IMSMQQueueInfo2` interfaces.

<47> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` or `IMSMQQueueInfo2` interfaces.

<48> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` or `IMSMQQueueInfo2` interfaces.

<49> [Section 3.10.4.1:](#) This method is not available in the `IMSMQQueueInfo` or `IMSMQQueueInfo2` interfaces.

[<50> Section 3.10.4.1.26:](#) Windows clients obtain the reference to the user SID by calling the Windows SDK function GetTokenInformation.

[<51> Section 3.11.4.1:](#) IMSMQQueueInfo3, IMSMQQueueInfo2, and IMSMQQueueInfo interfaces are returned for the IMSMQQueue3, IMSMQQueue2 and IMSMQQueue versions of this interface respectively.

[<52> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue interface.

[<53> Section 3.11.4.1:](#) IMSMQMessage3 and IMSMQMessage2 interfaces pointers are returned for the IMSMQQueue3 and IMSMQQueue2 versions of this interface respectively.

[<54> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue interface.

[<55> Section 3.11.4.1:](#) IMSMQMessage3 and IMSMQMessage2 interfaces pointers are returned for the IMSMQQueue3 and IMSMQQueue2 versions of this interface respectively.

[<56> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue interface.

[<57> Section 3.11.4.1:](#) IMSMQMessage3 and IMSMQMessage2 interfaces pointers are returned for the IMSMQQueue3 and IMSMQQueue2 versions of this interface respectively.

[<58> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue interface.

[<59> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue interface.

[<60> Section 3.11.4.1:](#) IMSMQMessage3 and IMSMQMessage2 interfaces pointers are returned for the IMSMQQueue3 and IMSMQQueue2 versions of this interface respectively.

[<61> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue interface.

[<62> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<63> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<64> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<65> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<66> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<67> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<68> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<69> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<70> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<71> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<72> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<73> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<74> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<75> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<76> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<77> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<78> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<79> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<80> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<81> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<82> Section 3.11.4.1:](#) An IMSMQMessage3 interfaces pointers is returned for the IMSMQQueue3 version of this interface respectively.

[<83> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<84> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue or IMSMQQueue2 interfaces.

[<85> Section 3.11.4.1:](#) This method is not available in the IMSMQQueue, IMSMQQueue2, or IMSMQQueue3 interfaces.

[<86> Section 3.13.4.1:](#) The ppqinfos parameter type is IMSMQQueueInfos3, IMSMQQueueInfos2 and IMSMQQueueInfos rather than IMSMQQueueInfos4 for interfaces IMSMQQuery3, IMSMQQuery2 and IMSMQQuery respectively.

[<87> Section 3.13.4.1:](#) This method is not available in the IMSMQQuery interface.

[<88> Section 3.13.4.1:](#) This method is not available in the IMSMQQuery or IMSMQQuery2 interfaces.

[<89> Section 3.13.4.1:](#) The ppqinfos parameter type is IMSMQQueueInfos3 rather than IMSMQQueueInfos4 for interface IMSMQQuery3.

<90> [Section 3.14.4.1:](#) The ppqinfoNext parameter type is IMSMQQueueInfos3, IMSMQQueueInfos2 and IMSMQQueueInfos rather than IMSMQQueueInfos4 for interfaces IMSMQQueueInfos3, IMSMQQueueInfos2 and IMSMQQueueInfos respectively.

<91> [Section 3.14.4.1:](#) This method is not available in the IMSMQQueueInfos interface.

<92> [Section 3.16.4.1:](#) This method is not available in the IMSMQEvent interface.

<93> [Section 3.16.4.2:](#) **IMSMQPrivateEvent** is used by Windows as an intermediate communication step between the client and an MSMQQueue instance. There is no requirement for a different implementation to do this and the same functionality could be achieved by the MSMQQueue object directly invoking the **DMSMQEventEvents::Arrived** and **DMSMQEventEvents::ArrivedError** methods.

<94> [Section 3.16.4.2.1:](#) Windows servers return the value of an internal handle which is only meaningful within the server process. Therefore, the value returned by this method is not useful in networked scenarios, and clients MUST ignore the returned value.

<95> [Section 3.16.4.3.2:](#) The Windows API documentation for MSMQEvent.ArrivedError indicates that the ErrorCode field contains the error code returned from MQReceiveMessage, see [\[MSDN-MQCOC\]](#). The possible error codes are listed below. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
MQ_ERROR_ACCESS_DENIED	0xC00E0025
MQ_ERROR_BUFFER_OVERFLOW	0xC00E001A
MQ_ERROR_DTC_CONNECT	0xC00E004C
MQ_ERROR_FORMATNAME_BUFFER_TOO_SMALL	0xC00E001F
MQ_ERROR_ILLEGAL_CURSOR_ACTION	0xC00E001C
MQ_ERROR_INSUFFICIENT_PROPERTIES	0xC00E003F
MQ_ERROR_INVALID_HANDLE	0xC00E0007
MQ_ERROR_IO_TIMEOUT	0xC00E001B
MQ_ERROR_LABEL_BUFFER_TOO_SMALL	0xC00E0063
MQ_ERROR_PROPERTY	0xC00E0002
MQ_ERROR_QUEUE_DELETED	0xC00E005A
MQ_ERROR_SENDER_CERT_BUFFER_TOO_SMALL	0xC00E002B
MQ_ERROR_SENDERID_BUFFER_TOO_SMALL	0xC00E0022
MQ_ERROR_STALE_HANDLE	0xC00E0056
MQ_ERROR_SYMM_KEY_BUFFER_TOO_SMALL	0xC00E0061
MQ_ERROR_TRANSACTION_USAGE	0xC00E0050
MQ_INFORMATION_OPERATION_PENDING	0x400E0006

Name	Value
MQ_INFORMATION_PROPERTY	0x400E0001

<96> [Section 3.16.4.4.1:](#) The Windows API documentation for `IConnectionPoint.GetConnectionInterface` includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
E_POINTER	0x80000005

<97> [Section 3.16.4.4.2:](#) The Windows API documentation for `IConnectionPoint.GetConnectionPointContainer` includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
E_POINTER	0x80000005

<98> [Section 3.16.4.4.3:](#) The Windows API documentation for `IConnectionPoint.Advise` includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
E_POINTER	0x80000005
CONNECT_E_ADVISELIMIT	0x80040202
CONNECT_E_CANNOTCONNECT	0x80040203

<99> [Section 3.16.4.4.4:](#) The Windows API documentation for `IConnectionPoint.Unadvise` includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
CONNECT_E_NOCONNECTION	0x80040201

<100> [Section 3.16.4.4.5:](#) The Windows API documentation for `IConnectionPoint.EnumConnections` includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
E_POINTER	0x80000005
CONNECT_E_ADVISELIMIT	0x80040202
CONNECT_E_CANNOTCONNECT	0x80040203

<101> [Section 3.16.4.5.1:](#) The Windows API documentation for `IConnectionPointContainer.EnumConnectionPoints` includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
E_POINTER	0x80000005

<102> [Section 3.16.4.5.2:](#) The Windows API documentation for `IConnectionPointContainer.FindConnectionPoint` includes the following error codes. For descriptions of the following error code name/value pairs, see [\[MS-MQMQ\]](#) section 2.7 and [\[MSDN-MQEIC\]](#), in that order.

Name	Value
S_OK	0x00000000
E_POINTER	0x80000005
CONNECT_E_NOCONNECTION	0x80040201

<103> [Section 3.17.3:](#) The MSMQ on all versions of Windows sets this to (0xFFFFFFFF).

<104> [Section 3.17.4.1:](#) Returns an `IMSMQQueueInfo3`, `IMSMQQueueInfo2` and `IMSMQQueueInfo` interface pointers rather than an `IMSMQQueueInfo4` interface pointer for interfaces `IMSMQMessage3`, `IMSMQMessage2` and `IMSMQMessage` respectively.

<105> [Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage` interface.

<106> [Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage` interface.

<107> [Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage` interface.

<108> [Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage` interface.

<109> [Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage` interface.

<110> [Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage` interface.

<111> [Section 3.17.4.1:](#) Returns an `IMSMQQueueInfo3` and `IMSMQQueueInfo2` interface pointers rather than an `IMSMQQueueInfo4` interface pointer for interfaces `IMSMQMessage3` and `IMSMQMessage2` respectively.

<112> [Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage` interface.

[<113> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<114> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<115> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<116> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<117> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<118> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<119> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<120> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<121> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<122> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<123> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<124> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<125> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<126> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<127> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<128> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<129> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<130> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<131> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage interface.

[<132> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage2 or IMSMQMessage interfaces.

[<133> Section 3.17.4.1:](#) Returns an IMSMQQueueInfo3 rather than an IMSMQQueueInfo4 interface pointer for the IMSMQMessage3 interface.

[<134> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage2 or IMSMQMessage interfaces.

[<135> Section 3.17.4.1:](#) Returns an IMSMQQueueInfo3 rather than an IMSMQQueueInfo4 interface pointer for the IMSMQMessage3 interface.

[<136> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage2 or IMSMQMessage interfaces.

[<137> Section 3.17.4.1:](#) Returns an IMSMQQueueInfo3 rather than an IMSMQQueueInfo4 interface pointer for the IMSMQMessage3 interface.

[<138> Section 3.17.4.1:](#) This method is not available in the IMSMQMessage2 or IMSMQMessage interfaces.

[<139> Section 3.17.4.1:](#) Returns an `IMSMQQueueInfo3` rather than an `IMSMQQueueInfo4` interface pointer for the `IMSMQMessage3` interface.

[<140> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<141> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<142> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<143> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<144> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<145> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<146> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<147> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<148> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<149> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<150> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<151> Section 3.17.4.1:](#) This method is not available in the `IMSMQMessage2` or `IMSMQMessage` interfaces.

[<152> Section 3.17.4.1.1:](#) All versions of Windows provide an implementation for this method for legacy behavior. The method functionality is identical to the get [IMSMQMessage4::MsgClass \(section 3.17.4.1.65\)](#) method.

[<153> Section 3.17.4.1.2:](#) Previous versions of MSMQ on the Windows platform may return `MQMSG_PRIV_LEVEL_BODY` (0x00000002), but this has since been deprecated and replaced with `MQMSG_PRIV_LEVEL_BODYBASE` (0x00000001), which indicates the same behavior.

[<154> Section 3.17.4.1.38:](#) The subset of supported values defined by the Message Queuing Binary Protocol Specification are contained in [\[MS-MQOB\]](#) section 2.2.2.4.

[<155> Section 3.17.4.1.39:](#) The subset of supported values defined by the Message Queuing Binary Protocol Specification are contained in [\[MS-MQOB\]](#) section 2.2.2.4.

[<156> Section 3.17.4.1.48:](#) Windows clients obtain the reference to the user SID by calling the Windows SDK function `GetTokenInformation`.

[<157> Section 3.17.4.1.49:](#) Windows clients obtain the reference to the user SID by calling the Windows SDK function `GetTokenInformation`.

8 Index

[NewEnum method](#)

A

Abort method ([section 3.8.4.1.2](#), [section 3.9.4.1.3](#))

Abstract data model

[ITransaction implementation class](#)

[MSMQApplication coclass](#)

[MSMQCollection coclass](#)

[MSMQCoordinatedTransactionDispenser coclass](#)

[MSMQDestination coclass](#)

[MSMQEvent coclass](#)

[MSMQManagement coclass](#)

[MSMQMessage coclass](#)

[MSMQOutgoingQueueManagement coclass](#)

[MSMQQuery coclass](#)

[MSMQQueue coclass](#)

[MSMQQueueInfo coclass](#)

[MSMQQueueInfos coclass](#)

[MSMQQueueManagement coclass](#)

[MSMQTransaction coclass](#)

[MSMQTransactionDispenser coclass](#)

[overview](#)

[Access method](#)

Ack method ([section 3.17.4.1.28](#), [section 3.17.4.1.29](#))

[ActiveQueues method](#)

AdminQueueInfo method ([section 3.17.4.1.24](#), [section 3.17.4.1.78](#), [section 3.17.4.1.79](#))

[AdminQueueInfo_v1 method](#)

AdminQueueInfo_v2 method ([section 3.17.4.1.73](#), [section 3.17.4.1.74](#))

ADsPath method ([section 3.10.4.1.39](#), [section 3.12.4.1.6](#), [section 3.12.4.1.7](#))

[Advise method](#)

[Applicability](#)

AppSpecific method ([section 3.17.4.1.17](#), [section 3.17.4.1.18](#))

[Arrived method](#)

[ArrivedError method](#)

[ArrivedTime method](#)

[AttachCurrentSecurityContext method](#)

[AttachCurrentSecurityContext2 method](#)

Authenticate method ([section 3.10.4.1.21](#), [section 3.10.4.1.22](#))

AuthenticationProviderName method ([section 3.17.4.1.62](#), [section 3.17.4.1.63](#))

AuthenticationProviderType method ([section 3.17.4.1.60](#), [section 3.17.4.1.61](#))

AuthLevel method ([section 3.17.4.1.4](#), [section 3.17.4.1.5](#))

B

BasePriority method ([section 3.10.4.1.17](#), [section 3.10.4.1.18](#))

BeginTransaction method ([section 3.6.4.1.1](#), [section 3.7.4.1.1](#))

Body method ([section 3.17.4.1.21](#), [section 3.17.4.1.22](#))

[BodyLength method](#)

[BytesInAllQueues method](#)

[BytesInJournal method](#)

[BytesInQueue method](#)

C

[Capability negotiation](#)

[Class method](#)

Close method ([section 3.11.4.1.6](#), [section 3.12.4.1.2](#))

Commit method ([section 3.8.4.1.1](#), [section 3.9.4.1.2](#))

[Common data types](#)

[CompoundMessage method](#)

[Connect method](#)

ConnectorTypeGuid method ([section 3.17.4.1.53](#), [section 3.17.4.1.54](#))

CorrelationId method ([section 3.17.4.1.26](#), [section 3.17.4.1.27](#))

[Count method](#)

[Create method](#)

[CreateTime method](#)

D

Data model - abstract

[ITransaction implementation class](#)

[MSMQApplication coclass](#)

[MSMQCollection coclass](#)

[MSMQCoordinatedTransactionDispenser coclass](#)

[MSMQDestination coclass](#)

[MSMQEvent coclass](#)

[MSMQManagement coclass](#)

[MSMQMessage coclass](#)

[MSMQOutgoingQueueManagement coclass](#)

[MSMQQuery coclass](#)

[MSMQQueue coclass](#)

[MSMQQueueInfo coclass](#)

[MSMQQueueInfos coclass](#)

[MSMQQueueManagement coclass](#)

[MSMQTransaction coclass](#)

[MSMQTransactionDispenser coclass](#)

[Data types](#)

[Delete method](#)

Delivery method ([section 3.17.4.1.7](#), [section 3.17.4.1.8](#))

[Destination method](#)

[DestinationQueueInfo method](#)

Destinations method ([section 3.12.4.1.12](#), [section 3.12.4.1.13](#))

DestinationSymmetricKey method ([section 3.17.4.1.56](#), [section 3.17.4.1.57](#))

[DirectoryServiceServer method](#)

[Disconnect method](#)

E

[EnableNotification method](#)

EncryptAlgorithm method ([section 3.17.4.1.38](#), [section 3.17.4.1.39](#))

[EnumConnectionPoints method](#)

[EnumConnections method](#)

[EodGetReceiveInfo method](#)

[EodGetSendInfo method](#)

[EodResend method](#)

[Examples - overview](#)

[Extension method](#) ([section 3.17.4.1.51](#), [section 3.17.4.1.52](#))

F

[Fields - vendor-extensible](#)

[FindConnectionPoint method](#)

[FireArrivedErrorEvent method](#)

[FireArrivedEvent method](#)

[ForeignStatus method](#)

[FormatName method](#) ([section 3.3.4.1.2](#), [section 3.10.4.1.8](#), [section 3.10.4.1.9](#), [section 3.12.4.1.10](#), [section 3.12.4.1.11](#))

[Full IDL](#)

G

[GetConnectionInterface method](#)

[GetConnectionPointContainer method](#)

[GetTransactionInfo method](#)

[Glossary](#)

H

[Handle method](#) ([section 3.11.4.1.4](#), [section 3.12.4.2.1](#), [section 3.12.4.2.2](#))

[Handle2 method](#)

[HashAlgorithm method](#) ([section 3.17.4.1.36](#), [section 3.17.4.1.37](#))

[Hwnd method](#)

I

[IADs method](#) ([section 3.12.4.1.4](#), [section 3.12.4.1.5](#))

[Id method](#)

[IDL](#)

[Implementer - security considerations](#)

[Index of security parameters](#)

[Informative references](#)

[Init method](#)

[Initialization](#)

[ITransaction implementation class](#)

[MSMQApplication coclass](#)

[MSMQCollection coclass](#)

[MSMQCoordinatedTransactionDispenser coclass](#)

[MSMQEvent coclass](#)

[MSMQManagement coclass](#)

[MSMQMessage coclass](#)

[MSMQOutgoingQueueManagement coclass](#)

[MSMQQuery coclass](#)

[MSMQQueue coclass](#)

[MSMQQueueDestination coclass](#)

[MSMQQueueInfo coclass](#)

[MSMQQueueInfos coclass](#)

[MSMQQueueManagement coclass](#)

[MSMQTransaction coclass](#)

[MSMQTransactionDispenser coclass](#)
[overview](#)

[InitNew method](#)

[Introduction](#)

[IsAuthenticated method](#)

[IsAuthenticated2 method](#)

[IsConnected method](#)

[IsDsEnabled method](#)

[IsFirstInTransaction method](#)

[IsFirstInTransaction2 method](#)

[IsLastInTransaction method](#)

[IsLastInTransaction2 method](#)

[IsLocal method](#)

[IsOpen method](#) ([section 3.11.4.1.5](#), [section 3.12.4.1.3](#))

[IsOpen2 method](#)

[IsTransactional method](#)

[IsTransactional2 method](#)

[IsWorldReadable method](#) ([section 3.10.4.1.25](#), [section 3.10.4.1.36](#))

[Item method](#)

[ITransaction implementation class](#)

[abstract data model](#)

[initialization](#)

[local events](#)

[message processing](#)

[overview](#)

[sequencing rules](#)

[timer events](#)

[timers](#)

[ITransaction method](#)

J

[Journal method](#) ([section 3.10.4.1.13](#), [section 3.10.4.1.14](#), [section 3.17.4.1.13](#), [section 3.17.4.1.14](#))

[JournalMessageCount method](#)

[JournalQuota method](#) ([section 3.10.4.1.23](#), [section 3.10.4.1.24](#))

L

[Label method](#) ([section 3.10.4.1.4](#), [section 3.10.4.1.5](#), [section 3.17.4.1.30](#), [section 3.17.4.1.31](#))

[Local events](#)

[ITransaction implementation class](#)

[MSMQApplication coclass](#)

[MSMQCollection coclass](#)

[MSMQCoordinatedTransactionDispenser coclass](#)

[MSMQDestination coclass](#)

[MSMQEvent coclass](#)

[MSMQManagement coclass](#)

[MSMQMessage coclass](#)

[MSMQOutgoingQueueManagement coclass](#)

[MSMQQuery coclass](#)

[MSMQQueue coclass](#)

[MSMQQueueInfo coclass](#)

[MSMQQueueInfos coclass](#)

[MSMQQueueManagement coclass](#)

[MSMQTransaction coclass](#)

[MSMQTransactionDispenser coclass
overview](#)
[LookupId method](#)
[LookupQueue method](#)
[LookupQueue v2 method](#)

M

Machine method ([section 3.2.4.3.6](#), [section 3.2.4.3.7](#),
[section 3.3.4.1.3](#))
[MachineIdOfMachineName method](#)
[MachineNameOfMachineId method](#)
MaxTimeToReachQueue method ([section 3.17.4.1.32](#),
[section 3.17.4.1.33](#))
MaxTimeToReceive method ([section 3.17.4.1.34](#),
[section 3.17.4.1.35](#))
Message processing
[ITransaction implementation class](#)
[MSMQApplication coclass](#)
[MSMQCollection coclass](#)
[MSMQCoordinatedTransactionDispenser coclass](#)
[MSMQEvent coclass](#)
[MSMQManagement coclass](#)
[MSMQMessage coclass](#)
[MSMQOutgoingQueueManagement coclass](#)
[MSMQQuery coclass](#)
[MSMQQueue coclass](#)
[MSMQQueueDestination coclass](#)
[MSMQQueueInfo coclass](#)
[MSMQQueueInfos coclass](#)
[MSMQQueueManagement coclass](#)
[MSMQTransaction coclass](#)
[MSMQTransactionDispenser coclass
overview](#)
[MessageCount method](#)
Messages
[overview](#)
[transport](#)
[ModifyTime method](#)
[MQACCESS enumeration](#)
[MQAUTHENTICATE enumeration](#)
[MQCALG enumeration](#)
[MQJOURNAL enumeration](#)
[MQMSGACKNOWLEDGEMENT enumeration](#)
[MQMSGAUTHENTICATION enumeration](#)
[MQMSGAUTHLEVEL enumeration](#)
[MQMSGCLASS enumeration](#)
[MQMSGCURSOR enumeration](#)
[MQMSGDELIVERY enumeration](#)
[MQMSGJOURNAL enumeration](#)
[MQMSGPRIVLEVEL enumeration](#)
[MQMSGSENDERIDTYPE enumeration](#)
[MQMSGTRACE enumeration](#)
[MQPRIVLEVEL enumeration](#)
[MQSHARE enumeration](#)
[MQTRANSACTION enumeration](#)
[MQTRANSACTIONAL enumeration](#)
MsgClass method ([section 3.17.4.1.65](#), [section
3.17.4.1.66](#))
MSMQApplication coclass
[abstract data model](#)

[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQCollection coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQCoordinatedTransactionDispenser coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQDestination coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQEvent coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQManagement coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQMessage coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQOutgoingQueueManagement coclass

[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQQuery coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQQueue coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQQueueInfo coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQQueueInfos coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQQueueManagement coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)

MSMQTransaction coclass
[abstract data model](#)
[initialization](#)
[local events](#)
[message processing](#)
[overview](#)
[sequencing rules](#)
[timer events](#)
[timers](#)
[MSMQVersionBuild](#) method
[MSMQVersionMajor](#) method
[MSMQVersionMinor](#) method
[MulticastAddress](#) method ([section 3.10.4.1.37](#), [section 3.10.4.1.38](#))

N

[Next](#) method
[NextHops](#) method
[Normative references](#)

O

[Open](#) method ([section 3.10.4.1.28](#), [section 3.12.4.1.1](#))
[Overview](#) (synopsis)

P

[Parameters - security index](#)
[PathName](#) method ([section 3.10.4.1.6](#), [section 3.10.4.1.7](#), [section 3.12.4.1.8](#), [section 3.12.4.1.9](#))
[PathNameDNS](#) method
[Pause](#) method
[Peek](#) method
[Peek v1](#) method
[PeekByLookupId](#) method
[PeekCurrent](#) method
[PeekCurrent v1](#) method
[PeekFirstByLookupId](#) method
[PeekLastByLookupId](#) method
[PeekNext](#) method
[PeekNext v1](#) method
[PeekNextByLookupId](#) method
[PeekPreviousByLookupId](#) method
[Preconditions](#)
[Prerequisites](#)
[Priority](#) method ([section 3.17.4.1.11](#), [section 3.17.4.1.12](#))
[PrivateQueues](#) method
[PrivLevel](#) method ([section 3.10.4.1.11](#), [section 3.10.4.1.12](#), [section 3.17.4.1.2](#), [section 3.17.4.1.3](#))
[Properties](#) method ([section 3.2.4.2.7](#), [section 3.6.4.1.2](#), [section 3.7.4.1.2](#), [section 3.9.4.2.2](#), [section 3.10.4.1.32](#), [section 3.11.4.1.19](#), [section 3.12.4.1.14](#), [section 3.13.4.1.2](#), [section 3.14.4.1.3](#), [section 3.16.4.1.1](#), [section 3.17.4.1.67](#))
[Purge](#) method

Q

[QUEUE_STATE](#) enumeration

[QueueGuid method](#)
[QueueInfo method](#)
[QueueType method](#)
Quota method ([section 3.10.4.1.15](#), [section 3.10.4.1.16](#))

R

[Receive method](#)
[Receive_v1 method](#)
[ReceiveByLookupId method](#)
[ReceiveByLookupIdAllowPeek method](#)
[ReceiveCurrent method](#)
[ReceiveCurrent_v1 method](#)
[ReceivedAuthenticationLevel method](#)
[ReceiveFirstByLookupId method](#)
[ReceiveLastByLookupId method](#)
[ReceiveNextByLookupId method](#)
[ReceivePreviousByLookupId method](#)

References

[informative](#)
[normative](#)
[overview](#)

[Refresh method](#)
[RegisterCertificate method](#)
[Relationship to other protocols](#)
[RELOPS enumeration](#)
Reset method ([section 3.11.4.1.10](#), [section 3.14.4.1.1](#))
ResponseDestination method ([section 3.17.4.1.80](#), [section 3.17.4.1.81](#))
ResponseQueueInfo method ([section 3.17.4.1.76](#), [section 3.17.4.1.77](#))
ResponseQueueInfo_v1 method ([section 3.17.4.1.15](#), [section 3.17.4.1.16](#))
ResponseQueueInfo_v2 method ([section 3.17.4.1.71](#), [section 3.17.4.1.72](#))
[Resume method](#)

S

Security

[implementer considerations](#)
[overview](#)
[parameter index](#)

Security method ([section 3.10.4.1.33](#), [section 3.10.4.1.34](#))
[Send method](#)
SenderCertificate method ([section 3.17.4.1.43](#), [section 3.17.4.1.44](#))
SenderId method ([section 3.17.4.1.45](#), [section 3.17.4.1.64](#))
SenderIdType method ([section 3.17.4.1.46](#), [section 3.17.4.1.47](#))
[SenderVersion method](#)
[SentTime method](#)

Sequencing rules

[ITransaction implementation class](#)
[MSMQApplication coclass](#)
[MSMQCollection coclass](#)
[MSMQCoordinatedTransactionDispenser coclass](#)
[MSMQEvent coclass](#)

[MSMQManagement coclass](#)
[MSMQMessage coclass](#)
[MSMQOutgoingQueueManagement coclass](#)
[MSMQQuery coclass](#)
[MSMQQueue coclass](#)
[MSMQQueueDestination coclass](#)
[MSMQQueueInfo coclass](#)
[MSMQQueueInfos coclass](#)
[MSMQQueueManagement coclass](#)
[MSMQTransaction coclass](#)
[MSMQTransactionDispenser coclass](#)
[overview](#)

ServiceTypeGuid method ([section 3.10.4.1.2](#), [section 3.10.4.1.3](#))

[ShareMode method](#)

Signature method ([section 3.17.4.1.58](#), [section 3.17.4.1.59](#))

[SoapBody method](#)
[SoapEnvelope method](#)
[SoapHeader method](#)
[SourceMachineGuid method](#)
[Standards assignments](#)
[State method](#)

T

[Tidy method](#)

Timer events

[ITransaction implementation class](#)
[MSMQApplication coclass](#)
[MSMQCollection coclass](#)
[MSMQCoordinatedTransactionDispenser coclass](#)
[MSMQDestination coclass](#)
[MSMQEvent coclass](#)
[MSMQManagement coclass](#)
[MSMQMessage coclass](#)
[MSMQOutgoingQueueManagement coclass](#)
[MSMQQuery coclass](#)
[MSMQQueue coclass](#)
[MSMQQueueInfo coclass](#)
[MSMQQueueInfos coclass](#)
[MSMQQueueManagement coclass](#)
[MSMQTransaction coclass](#)
[MSMQTransactionDispenser coclass](#)
[overview](#)

Timers

[ITransaction implementation class](#)
[MSMQApplication coclass](#)
[MSMQCollection coclass](#)
[MSMQCoordinatedTransactionDispenser coclass](#)
[MSMQDestination coclass](#)
[MSMQEvent coclass](#)
[MSMQManagement coclass](#)
[MSMQMessage coclass](#)
[MSMQOutgoingQueueManagement coclass](#)
[MSMQQuery coclass](#)
[MSMQQueue coclass](#)
[MSMQQueueInfo coclass](#)
[MSMQQueueInfos coclass](#)
[MSMQQueueManagement coclass](#)
[MSMQTransaction coclass](#)

[MSMQTransactionDispenser coclass](#)

[overview](#)

Trace method ([section 3.17.4.1.9](#), [section 3.17.4.1.10](#))

[Transaction method](#)

[TransactionalStatus method](#)

[TransactionId method](#)

[TransactionStatusQueueInfo method](#)

[Transport](#)

U

[Unadvise method](#)

[Update method](#)

V

[Vendor-extensible fields](#)

[Versioning](#)

W

[Windows behavior](#)

X

[XACTTC enumeration](#)