[MS-OXCMSG]:
Message and Attachment Object Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Revised and edited technical content.
10/01/2008	1.03		Revised and edited technical content.
12/03/2008	1.04		Updated IP notice.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	5.0.0	Major	Updated and revised the technical content.
05/05/2010	6.0.0	Major	Updated and revised the technical content.

Table of Contents

1 Introduction

The Message and Attachment Object Protocol provides the methods used within the server for manipulating **Message objects**.

1.1 Glossary

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

address book
ASCII
appointment
attachment
Attachment object
Archive Policy
Bcc recipient
Cc recipient
code page
contact
Contact object
contents table
Coordinated Universal Time (UTC)
download
Email object
Embedded Message object
EntryID
flags
folder
Folder Associated Information (FAI)
folder ID (FID)
Folder object
GUID
handle
header
HTML
identifier
Journal object
LogonID
Logon object
LongTermID
message
message class
message ID (MID)
Message object
metafile
MIME
named property
non-Unicode
Note object
permissions
plain text
primary recipient
property
property ID

property tag
public folder
read receipt
recipient
remote operation (ROP)
remote procedure call (RPC)
Retention Policy
Rich Text Format (RTF)
ROP request
ROP request buffer
ROP response buffer
S/MIME
soft delete
store
Store object
table
To recipient
Transport Neutral Encapsulation Format (TNEF)
Unicode
Uniform Resource Identifier (URI)

The following terms are specific to this document:

archive tag: An element that contains information about the **Archive Policy** of a **Message object** or **folder**.

clear signed body: A message body that was promoted from a clear-signed S/MIME message. For more information on clear-signed messages, see [MS-OXOSMIME] section 2.2.1.

header message object: A Message object that contains partial information about a message left on a server such as an identifier for the message, the display names of the recipients and sender of the message, the subject of the message, and the delivery time of the message. It allows a client to display enough information about a message to let a user choose which messages to download.

Object Linking and Embedding (OLE): A Microsoft compound document standard that enables cross-application linking and embedding of objects.

retention tag: An element that contains information about the **Retention Policy** of a **Message object** or **folder**.

undefined body: A body with no defined content.

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

1.2 References

1.2.1 Normative References

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

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference", March 2007, http://go.microsoft.com/fwlink/?LinkId=112265

[MS-OXCDATA] Microsoft Corporation, "Data Structures", April 2008.

[MS-OXCFOLD] Microsoft Corporation, "Folder Object Protocol Specification", April 2008.

[MS-OXCMAIL] Microsoft Corporation, "RFC2822 and MIME to E-Mail Object Conversion Protocol Specification", April 2008.

[MS-OXCPRPT] Microsoft Corporation, "Property and Stream Object Protocol Specification", April 2008.

[MS-OXCROPS] Microsoft Corporation, "Remote Operations (ROP) List and Encoding Protocol Specification", April 2008.

[MS-OXCSTOR] Microsoft Corporation, "Store Object Protocol Specification", April 2008.

[MS-OXCTABL] Microsoft Corporation, "Table Object Protocol Specification", April 2008.

[MS-OXGLOS] Microsoft Corporation, "Exchange Server Protocols Master Glossary", April 2008.

[MS-OXOABK] Microsoft Corporation, "Address Book Object Protocol Specification", April 2008.

[MS-OXOCAL] Microsoft Corporation, "Appointment and Meeting Object Protocol Specification", April 2008.

[MS-OXOCNTC] Microsoft Corporation, "Contact Object Protocol Specification", April 2008.

[MS-OXODOC] Microsoft Corporation, "Document Object Protocol Specification", April 2008.

[MS-OXOJRNL] Microsoft Corporation, "Journal Object Protocol Specification", April 2008.

[MS-OXOMSG] Microsoft Corporation, "E-Mail Object Protocol Specification", April 2008.

[MS-OXONOTE] Microsoft Corporation, "Note Object Protocol Specification", April 2008.

[MS-OXOPOST] Microsoft Corporation, "Post Object Protocol Specification", April 2008.

[MS-OXORSS] Microsoft Corporation, "RSS Object Protocol Specification", April 2008.

[MS-OXOSFLD] Microsoft Corporation, "Special Folders Protocol Specification", April 2008.

[MS-OXOSMIME] Microsoft Corporation, "S/MIME E-Mail Object Protocol Specification", April 2008.

[MS-OXOSMMS] Microsoft Corporation, "Short Message Service (SMS) and Multimedia Messaging Service (MMS) Object Protocol Specification", April 2008.

[MS-OXOTASK] Microsoft Corporation, "Task-Related Objects Protocol Specification", April 2008.

[MS-OXOUM] Microsoft Corporation, "Voice Mail and Fax Objects Protocol Specification", April 2008.

[MS-OXPROPS] Microsoft Corporation, "Exchange Server Protocols Master Property List", April 2008.

[MS-OXTNEF] Microsoft Corporation, "Transport Neutral Encapsulation Format (TNEF) Data Structure", April 2008.

[MS-WMF] Microsoft Corporation, "Windows Metafile Format Specification", June 2007, http://go.microsoft.com/fwlink/?LinkId=112205

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

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001, http://www.ietf.org/rfc/rfc2822.txt

1.2.2 Informative References

None.

1.3 Overview

1.3.1 Messages

Message objects are representations of end-users' data that store **properties** and are persisted in a **folder** hierarchy within a **message store**.

1.3.2 FAI Messages

Folder Associated Information (FAI) messages are a type of message that contains non-user data needed by the client or server. FAI messages are persisted in the same way as Message objects, but cannot be sent.

1.3.3 Message Recipients

Message objects allow clients to associate one or more **recipients** to a message. A recipient is a set of properties that describe where the message is to be delivered. Clients add, remove or modify recipients through the **RecipientRows**.

1.3.4 Message Attachments

An **Attachment object** is used by a client to associate files, **Object Linking and Embedding (OLE)** objects, other messages, or binary data with a particular Message object. Because Attachment objects are created, maintained, and accessed only in the context of a message, they are considered sub-objects. Operations that affect the location of a Message object also apply to its **attachments**. Clients retrieve information about attachments in a message via an attachment **table**, which is a table object as specified in the [MS-OXCTABL].

1.4 Relationship to Other Protocols

The Message and Attachment Object Protocol relies on folders, tables, and properties, as specified in [MS-OXCFOLD], [MS-OXOSFLD], [MS-OXCTABL], [MS-OXCPRPT], as well as the underlying **remote operations** transport, specified in [MS-OXCROPS].

At the time of this publication, the following protocols are known to extend Message and Attachment Object Protocol.

- Appointment and Meeting Object Protocol, [MS-OXOCAL]
- Contact Object Protocol, [MS-OXOCNTC]
- E-mail Object Protocol, [MS-OXOMSG]
- Task –Related Objects Protocol, [MS-OXOTASK]

- Note Object Protocol, [MS-OXONOTE]
- Journal Object Protocol, [MS-OXOJRNL]
- RSS Object Protocol, [MS-OXORSS]
- Post Object Protocol, [MS-OXOPOST]
- SMS and MMS Object Protocol, [MS-OXOSMMS]
- Document Object Protocol, [MS-OXODOC]
- S/MIME E-mail Object Protocol, [MS-OXOSMIME]
- Voice Mail and Fax Objects Protocol, [MS-OXOUM]

1.5 Prerequisites/Preconditions

The Message and Attachment Object Protocol assumes the client has previously logged on to the server and has acquired a **handle** to the **Folder object** upon which it needs to operate. For more information, see [MS-OXCFOLD] and [MS-OXCSTOR].

1.6 Applicability Statement

The Message and Attachment Object Protocol can be used as the basis for different types of personal information messages, like E-mail, **Contacts**, **Appointments**, Notes, etc.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

A third-party application can create its own set of **named properties** on a Message object as specified in [MS-OXCPRPT]. A third-party application can also extend the Message and Attachment Object Protocol to implement its own object type, by changing PidTagMessageClass. See [MS-OXONOTE] for a simple example that extends this protocol to implement an electronic representation of a "Sticky Note".

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The **ROP request buffers** and **ROP response buffers** specified by this protocol are sent to and respectively are received from the server using the underlying remote operations transport as specified in [MS-OXCROPS].

2.2 Message Syntax

Message objects can be created and modified by clients and servers. Except where noted below, this section defines constraints to which both clients and servers adhere when operating on Message objects.

Clients operate on Message objects using the ROPs as specified in section 2.2.3, and the Property and Stream Object Protocol. See [MS-OXCPRPT] for more details.

Unless otherwise specified below, all property constraints specified in [MS-OXPROPS] apply to Message objects. A Message object can also contain other properties defined in [MS-OXPROPS] but these properties have no impact on this protocol.

When a property is referred to as "read-only for the client" the server returns an error and ignore any request to change the value of that property.

2.2.1 Message Object Properties

All property value types (for example, **PtypBoolean** and **PtypInteger32**) that are specified in the following sub-sections are defined in [MS-OXCDATA] section 2.12.1.

2.2.1.1 General Properties

The following properties exist on all Message objects. These properties are read-only for the client. For specifications of the properties listed here see [MS-OXCPRPT].

PidTagAccess

PidTagAccessLevel

PidTagChangeKey

PidTagCreationTime

PidTagLastModificationTime

PidTagLastModifierName

PidTagObjectType

PidTagRecordKey

PidTagSearchKey

2.2.1.2 PidTagHasAttachments

Type: **PtypBoolean**.

Indicates whether the Message object contains at least one attachment. This property is read-only for the client.

The server computes this property from the **mfHasAttach** flag of PidTagMessageFlags.

2.2.1.3 PidTagMessageClass

Type: **PtypString**.

Denotes the specific type of the Message object. It determines the set of properties defined for the message, the kind of information the message conveys, and how to handle the message.

All characters in this property MUST be from the **ASCII** characters 0x20 through 0x7F. It MUST NOT end with a period (ASCII character 0x2E), and its length MUST be greater than zero and less than 256 characters. Furthermore, its length SHOULD be fewer than 128 characters, because some operations require extending the value of PidTagMessageClass.

The value of this property is interpreted in groups of characters separated by periods ("."). Each group specifies a derived type of object. If a server or client does not recognize a **message class**, it reverts to acting on all but the last group, recursively, until a recognized form remains. A message class of "IPM.Note" denotes a standard Message object, and a message class of "Remote. IPM.Note" indicates a **header message object**.

2.2.1.4 PidTagMessageCodepage

Type: **PtypInteger32**, unsigned

Specifies the **code page** used to encode the non-**Unicode string properties** on this Message object. The Folder Object code page is used if this property is set to "0x0000".

2.2.1.5 PidTagMessageLocaleId

Type: **PtypInteger32**, unsigned

Contains the Windows LCID of the end-user who created this message. For more details see [MS-LCID].

2.2.1.6 PidTagMessageFlags

Type: **PtypeInteger32**

Specifies the status of the Message object. Set to a bitwise OR of zero or more of the values from the following tables.

After the first successful RopSaveChangesMessage, as described in section 2.2.3.3, these **flags** are read-only for the client.

Name	Value	Description
mfRead	0x00000001	The message is marked as having been read.
mfUnsent	0x00000008	The message is still being composed. This bit is cleared by the server when responding to RopSubmitMessage with a success code. See [MS-OXOMSG] for details.
mfResend	0x00000080	The message includes a request for a resend operation with a non-delivery

Name	Value	Description
		report. See [MS-OXOMSG] for details.

These flags are always read-only for the client.

Type	Value	Description
mfUnmodified	0x00000002	The message has not been modified since it was first saved (if unsent) or it was delivered (if sent).
mfSubmitted	0x00000004	The message is marked for sending as a result of a call to RopSubmitMessage [MS-OXOMSG].
mfHasAttach	0x00000010	The message has at least one attachment. This flag corresponds to the message's PidTagHasAttachments property.
mfFromMe	0x00000020	The user receiving the message was also the user who sent the message.
mfFAI	0x00000040	The message is an FAI message.
mfNotifyRead	0x00000100	The user who sent the message has requested notification when a recipient first reads it.
mfNotifyUnread	0x00000200	The user who sent the message has requested notification when a recipient deletes it before reading or the Message object expires as specified in [MS-OXOMSG].
mfInternet	0x00002000	The incoming message arrived over the Internet and originated either outside the organization or from a source the gateway does not consider trusted.
mfUntrusted	0x00008000	The incoming message arrived over an external link other than X.400 or the Internet. It originated either outside the organization or from a source the gateway does not consider trusted.

PidTagMessageFlags are also modified using RopSetMessageReadFlag request, as described in section 2.2.3.11, or RopSetReadFlags, as described in section 2.2.3.10.

2.2.1.7 PidTagMessageSize

Type: **PtypInteger32**, unsigned

Contains the size in bytes consumed by the Message object on the server. This property is read-only for the client.

2.2.1.8 PidTagMessageStatus

Type: **PtypInterger32**

Specifies the status of a message in a contents table. Contains a bitwise OR of zero or more of the following values.

Type	Value	Description
msRemoteDownload	0x00001000	The message has been marked for **downloading** from the remote

Type	Value	Description
		message store to the local client.
msInConflict	0x00000800	This is a conflict resolve message as specified in [MS-OXCSYNC]. This is a read-only value for the client.
msRemoteDelete	0x00002000	The message has been marked for deletion at the remote message store without downloading to the local client.

In order to set PidTagMessageStatus, the client does not include it in a RopSetProperties request, as described in [MS-OXCPRPT]. Instead, the client calls RopSetMessageStatus, as described in section 2.2.3.8.

In order to get PidTagMessageStatus, the client does not include it in a RopGetPropertiesSpecific request. Instead, the client calls RopSetMessageStatus, as described in section 2.2.3.8. Additionally, PidTagMessageStatus can be set as a column on a **contents table** (for more details on tables, see [MS-OXCTABL]).

2.2.1.9 PidTagSubjectPrefix

Type: **PtypString**

Contains the prefix for the subject of the message. Set by the client, but can be an empty string if there is no subject. The sum of the lengths of PidTagNormalizedSubject and PidTagSubjectPrefix MUST be less than 254 characters.

The client does not include PidTagSubjectPrefix in a RopGetPropertiesSpecific request. Instead, the client uses the **SubjectPrefix** field from the RopOpenMessage response buffer.

2.2.1.10 PidTagNormalizedSubject

Type: **PtypString**

Contains the normalized subject of the message. Set by the client, but can be an empty string if there is no subject. The sum of the lengths of PidTagNormalizedSubject and PidTagSubjectPrefix MUST be less than 254 characters.

The client does not include PidTagNormalizedSubject in a RopGetPropertiesSpecific request. Instead, the client uses the **NormalizedSubject** field from the RopOpenMessage response buffer.

2.2.1.11 PidTagImportance

Type: **PtypInteger32**

Indicates the level of importance assigned by the end user to the Message object; MUST be set to one of the following values.

Value	Description
0x00000000	Low importance.
0x00000001	Normal importance.
0x00000002	High importance.

2.2.1.12 PidTagPriority

Type: **PtypInteger32**

Indicates the client's request for the priority at which the message is to be sent by the messaging system. Is one of the following values.

Value	Description
0x00000001	Urgent.
0x00000000	Normal.
0xFFFFFFFF	Non-Urgent

2.2.1.13 PidTagSensitivity

Type: **PtypInteger32**

Indicates the sender's assessment of the sensitivity of the Message object. Is one of the following.

Value	Description
0x00000000	Normal
0x00000001	Personal
0x00000002	Private
0x00000003	Confidential

2.2.1.14 PidLidSmartNoAttach

Type: **PtypBoolean**

Indicates whether the Message object has no end-user visible attachments. If this property is unset, then a default value of "0x00" is assumed.

2.2.1.15 PidLidPrivate

Type: **PtypBoolean**

Indicates whether the end-user wishes for this Message object to be hidden from other users who have access to the Message object.

2.2.1.16 PidLidSideEffects

Type: **PtypInteger32**

Controls how a Message object is handled by the client when acting on end-user input. Is set to a bitwise OR of zero or more of the following flags.

Name	Value	Description
seOpenToDelete	0x0001	Additional processing is required on the Message object when deleting.

Name	Value	Description
seNoFrame	0x0008	No UI is associated with the Message object.
seCoerceToInbox	0x0010	Additional processing is required on the Message object when moving or copying to a Folder Object with a PidTagContainerClass of "IPF.Note". For more details about the PidTagContainerClass property, see [MS-OXOSFLD] section 2.2.5.
seOpenTocopy	0x0020	Additional processing is required on the Message object when copying to another folder.
seOpenToMove	0x0040	Additional processing is required on the Message object when moving to another folder.
seOpenForCtxMenu	0x0100	Additional processing is required on the Message object when displaying verbs to the end-user.
seCannotUndoDelete	0x0400	Cannot undo delete operation, MUST NOT be set unless "0x0001" is set.
seCannotUndoCopy	0x0800	Cannot undo copy operation, MUST NOT be set unless "0x0020" is set.
seCannotUndoMove	0x1000	Cannot undo move operation, MUST NOT be set unless "0x0040" is set.
seHasScript	0x2000	The Message object contains end-user script.
seOpenToPermDelete	0x4000	Additional processing is required to permanently delete the Message object.

2.2.1.17 PidNameKeywords

Type: **PtypMultipleString**

Contains keywords or categories for the Message object. The length of each string within the multi-value string is less than 256.

2.2.1.18 PidLidCommonStart

Type: **PtypTime**

Indicates the start time for the Message object. Is less than or equal to the value of PidLidCommonEnd. This time is interpreted as **Coordinated Universal Time (UTC)** or local depending on the specification of an extension to this protocol.

2.2.1.19 PidLidCommonEnd

Type: **PtypTime**

Indicates the end time for the Message object. MUST be greater than or equal to the value of PidLidCommonStart. This time is interpreted as UTC or local depending on the specification of an extension to this protocol.

2.2.1.20 Body Properties

A group of related properties valid on any Message object that specify the body text format and contents and conform to the specification in [MS-OXBBODY].

2.2.1.20.1 PidTagBody

Type: **PtypString**

Contains the unformatted text analogous to the text/plain body of [RFC2822].

2.2.1.20.2 PidTagNativeBody

Type: **PtypInteger32**

Indicates the best available format for storing the **message body**. The possible property values are shown in the following table.

Meaning	Value
Undefined body	0x00
Plain text body	0x01
RTF compressed body	0x02
HTML body	0x03
Clear signed body	0x04

2.2.1.20.3 PidTagBodyHtml

Type: **PtypBinary**

Contains the HTML body as specified in [RFC2822].

2.2.1.20.4 PidTagRtfCompressed

Type: **PtypBinary**

Contains a Rich Text Format (RTF) body compressed as specified in [MS-OXRTFCP].

2.2.1.20.5 PidTagRtfInSync

Type: **PtypBoolean**

Indicates whether the RTF body has been synchronized with the contents in PidTagBody.

2.2.1.20.6 PidTagInternetCodepage

Type: **PtypInteger32**

Indicates the code page used for PidTagBody or PidTagBodyHtml.

2.2.1.21 Contact Linking Properties

A group of related properties valid on any Message object containing information about the linked **Contact objects**.

2.2.1.21.1 PidLidContactLinkEntry

Type: **PtypBinary**

Contains the list of **Address Book EntryIDs** linked to by this Message object.

Size in bytes	Meaning	Notes
4	Address Entry COUNT	
4	Size of this property minus 4 (FieldSize)	
variable	Address Book EntryID data	Repeats Address Entry COUNT times
0 – 3	Padding to make FieldSize a multiple of 4	Each padding BYTE MUST be 0x00.

2.2.1.21.2 PidLidContacts

Type: **PtypMultipleString**

Contains the PidTagDisplayName of each address book EntryID referenced in the value of PidLidContactLinkEntry. Can also include names not referenced in PidLidContactLinkEntry.

2.2.1.21.3 PidLidContactLinkName

Type: **PtypString**

Contains the elements of PidLidContacts, separated by a semicolon and a space ("; ").

2.2.1.21.4 PidLidContactLinkSearchKey

Type: **PtypBinary**

Contains the list of search keys for the Contact object linked to by this Message object. Search keys are used to find related objects. Search keys for address book data are further specified by PidTagSearchKey in [MS-OXOABK].

Length in bytes	Meaning	Notes
2	**ContactEntryCount**	
variable	SearchKey data	Repeats **ContactEntryCount** times

2.2.1.22 Retention and Archive Properties

Properties that specify information about the **Retention Policy** or **Archive Policy**. These properties are valid on e-mail Message objects. Some of these properties are also valid on folders. The Retention Policy and the Archive Policy are independent features. The server can enable one of these policies, both of these policies, or neither of them. For details about how the Retention Policy and Archive Policy settings are communicated between client and server, see [MS-OXOCFG].

2.2.1.22.1 PidTagArchiveTag

Type: **PtypBinary**

Specifies the **GUID** of an **archive tag**. The PidTagArchiveTag property can be present on both Message objects and folders and can be set by both client and server.

2.2.1.22.2 PidTagPolicyTag

Type: **PtypBinary**

Specifies the GUID of a **retention tag**. The PidTagPolicyTag property can be present on both Message objects and folders and can be set by both client and server.

2.2.1.22.3 PidTagRetentionPeriod

Type: **PtypInteger32**

Specifies the number of days that a Message object can be retained. The PidTagRetentionPeriod property can be present on both Message objects and folders and can be set by both client and server.

The presence of the PidTagRetentionPeriod property on a Message object indicates that the retention tag on that Message object was explicitly applied by the end-user. If the value of PidTagRetentionPeriod is 0, the Message object never expires.

When the PidTagRetentionPeriod property is present on a folder, it has no special significance; it simply specifies the retention period that corresponds to the retention tag on that folder.

2.2.1.22.4 PidTagStartDateEtc

Type: **PtypBinary**

A composite property that holds two pieces of information, as follows:

- Default retention period — The first four bytes contain the retention period of the default retention tag. (A default retention tag is applied to a Message object when a regular retention tag is not present on the Message object. The absence of a regular retention tag indicates that the Message object does not have a specific Retention Policy. The application of the default tag is based on the Message object's message class).
- Start date — The next eight bytes contain the date, in UTC, from which the Message object's age is calculated.

The PidTagStartDateEtc property can be present on only Message objects, not on folders. The PidTagStartDateEtc property is read-only for the client and is set by only the server.<1>

2.2.1.22.5 PidTagRetentionDate

Type: **PtypTime**

Specifies the date, in UTC, after which a Message object is expired by the server. The PidTagRetentionDate property can be present only on Message objects, not on folders. If the property is not present, the Message object never expires. The PidTagRetentionDate property can be set by both client and server.

The value of the PidTagRetentionDate property is calculated from the values of other properties. The values used in the calculation depend on whether the Message object has a specific Retention Policy. (A Message object will have the default Retention Policy in the absence of a specific Retention Policy.) The explicit method of calculation is as follows:

- When the Message object has a specific Retention Policy:

  PidTagRetentionDate = PidTagMessageDeliveryTime + PidTagRetentionPeriod. If PidTagMessageDeliveryTime does not exist, PidTagCreationTime is used.

- When the Message object has the default Retention Policy:

  PidTagRetentionDate = PidTagMessageDeliveryTime + default retention period. If PidTagMessageDeliveryTime does not exist, PidTagCreationTime is used.

2.2.1.22.6 PidTagRetentionFlags

Type: **PtypInteger32**

Contains flags that specify the status or nature of an item's retention tag or archive tag. The PidTagRetentionFlags property can be present on both Message objects and folders and can be set by both client and server.

The value of the PidTagRetentionFlags property is a bitwise OR of zero or more of the values from the following table:

Name	Value	Description
ExplicitTag	0x00000001	The retention tag on the folder is explicitly set.
UserOverride	0x00000002	The retention tag was not changed by the end-user.
AutoTag	0x00000004	The retention tag on the Message object is an auto-tag, which is predicted by the system.
PersonalTag	0x00000008	The retention tag on the folder is of a personal type and can be made available to the end-user.
ExplicitArchiveTag	0x000000010	The archive tag on the folder is explicitly set.
KeepInPlace	0x000000020	The Message object remains in place and is not archived.
SystemData	0x000000040	The Message object or folder is system data.

2.2.1.22.7 PidTagArchivePeriod

Type: **PtypInteger32**

Specifies the number of days that a Message object can remain unarchived. The PidTagArchivePeriod property can be present on both Message objects and folders and can be set by both client and server.

The presence of the PidTagArchivePeriod property on a Message object indicates that the archive tag on that Message object was explicitly applied by the end-user. If the value of PidTagArchivePeriod is 0, the Message object is never moved to archive by the server.

When the PidTagArchivePeriod property is present on a folder, it has no special significance; it simply specifies the archive period that corresponds to the archive tag on that folder.

2.2.1.22.8 PidTagArchiveDate

Type: **PtypTime**

Specifies the date, in UTC, after which a Message object is moved to archive by the server. The PidTagArchiveDate property can be present on only Message objects, not on folders, and can be set by both client and server. If the PidTagArchiveDate property is not present, the Message object is never moved to archive by the server.

The value of the PidTagArchiveDate property is calculated from the values of other properties as follows:

PidTagArchiveDate = start date + PidTagArchivePeriod

The start date is obtained from the last eight bytes of the PidTagStartDateEtc property.

2.2.2 Attachment Object Properties

All property value types (for example, **PtypBoolean** and **PtypInteger32**) that are specified in the following sub-sections are defined in [MS-OXCDATA] section 2.12.1.

2.2.2.1 General Properties

The following properties exist on any Attachment object. These properties are set by the server and are read-only for the client. For specifications of these properties see [MS-OXCPRPT].

PidTagAccessLevel

PidTagObjectType

PidTagRecordKey

2.2.2.2 PidTagLastModificationTime

Type: **PtypTime**, in UTC

Indicates the last time the file referenced by the Attachment object was modified, or the last time the Attachment object itself was modified.

2.2.2.3 PidTagCreationTime

Type: **PtypTime**, in UTC

Indicates the time the file referenced by the Attachment object was created, or the time the Attachment object itself was created.

2.2.2.4 PidTagDisplayName

Type: **PtypString**

Contains the name of the attachment as input by the end user. Set to the same value as PidTagAttachLongFilename.

2.2.2.5 PidTagAttachSize

Type: **PtypInteger32**, unsigned

Contains the size in bytes consumed by the Attachment object on the server. This property is read-only for the client.

2.2.2.6 PidTagAttachNumber

Type: **PtypInteger32**, unsigned

Identifies the Attachment object within its Message object. MUST be unique among the Attachment objects in a Message.

2.2.2.7 PidTagAttachDataBinary

Type: **PtypBinary**

Contains the contents of the file to be attached.

2.2.2.8 PidTagAttachDataObject

Type: **PtypObject**

Contains the binary representation of the Attachment object in an application-specific format.

2.2.2.9 PidTagAttachMethod

Type: **PtypInteger32**

Represents the way the contents of an attachment are accessed. Set to one of the following values.

Name	Value	Description
afNone	0x00000000	The attachment has just been created.
afByValue	0x00000001	PidTagAttachDataBinary contains the attachment data.
afByReference	0x00000002	PidTagAttachLongPathname contains a fully qualified path identifying the attachment **To recipients** with access to a common file server.
afByReferenceOnly	0x00000004	PidTagAttachLongPathname contains a fully qualified path identifying the attachment.
afEmbeddedMessage	0x00000005	The attachment is an embedded message that is accessed via RopOpenEmbeddedMessage
afStorage	0x00000006	PidTagAttachDataObject contains data in an application-specific format.

2.2.2.10 PidTagAttachLongFilename

Type: **PtypString**

Contains the full filename and extension of the Attachment object.

2.2.2.11 PidTagAttachFilename

Type: **PtypString**

Contains the 8.3 name of PidTagAttachLongFilename

2.2.2.12 PidTagAttachExtension

Type: **PtypString**

Contains a filename extension that indicates the document type of an attachment.

2.2.2.13 PidTagAttachLongPathname

Type: **PtypString**

Contains the fully qualified path and filename with extension.

2.2.2.14 PidTagAttachPathname

Type: **PtypString**

Contains the 8.3 name of PidTagAttachLongPathname.

2.2.2.15 PidTagAttachTag

Type: **PtypBinary**

Contains the identifier information for the application which supplied the Attachment object's data. This property can be left unset; if set, it MUST be one of the following.

Definition	Data	Comments
TNEF	{0x2A,86,48,86,F7,14,03,0A,01}	See [MS-OXTNEF]
afStorage	{0x2A,86,48,86,F7,14,03,0A,03,02,01}	Data is in an application-specific format.
MIME	{0x2A,86,48,86,F7,14,03,0A,04}	See [MS-OXCMAIL]

2.2.2.16 PidTagRenderingPosition

Type: **PtypInteger32**, unsigned

Represents an offset, in rendered characters, to use when rendering an attachment within the main message text. The value "0xFFFFFFFF" indicates a hidden attachment.

2.2.2.17 PidTagAttachRendering

Type: **PtypBinary**

Contains a Windows **metafile** as specified in [MS-WMF] for the Attachment object.

2.2.2.18 PidTagAttachFlags

Type: **PtypInteger32**, as a bit field

Indicates which body formats might reference this attachment when rendering data. Contains a bitwise OR of zero or more of the following flags. If this property is absent or its value is 0x00000000, then the attachment is available to be rendered in any format.

Value	Meaning
0x00000001	The Attachment object is not available to be rendered in HTML.
0x00000002	The Attachment object is not available to be rendered in Rich Text Format.
0x00000004	The Attachment object is referenced and rendered within the HTML body of the associated Message object. See PidTagBodyHtml.

2.2.2.19 PidTagAttachTransportName

Type: **PtypString**.

Contains the name of an attachment file, modified so that it can be correlated with TNEF messages, see [MS-OXTNEF].

2.2.2.20 PidTagAttachEncoding

Type: **PtypBinary**

Contains encoding information about the Attachment object. If the attachment is in MacBinary format, then this property is set to "{0x2A,86,48,86,F7,14,03,0B,01}"; otherwise, it is unset. <2> This property is used to indicate that the attachment content, which is the value of the PidTagAttachDataBinary property, MUST be encoded in the MacBinary format, as specified in [MS-OXCMAIL].

2.2.2.21 PidTagAttachAdditionalInformation

Type: **PtypString**

MUST be unset if PidTagAttachEncoding is unset. MUST be set to ":CREA:TYPE" if PidTagAttachEncoding is set.

2.2.2.22 PidTagAttachmentLinkId

Type: **PtypInteger32**

The type of Message object to which this attachment is linked. MUST be "0x00000000", unless overridden by other protocols that extend the Message and Attachment Object Protocol as noted in section 1.4.

2.2.2.23 PidTagAttachmentFlags

Type: **PtypInteger32**

Indicates special handling for this Attachment object. MUST be "0x00000000", unless overridden by other protocols that extend the Message and Attachment Object Protocol as noted in section 1.4

2.2.2.24 PidTagAttachmentHidden

Type: **PtypBoolean**

Indicates whether this Attachment object is hidden from the end user.

2.2.2.25 MIME properties

The following properties contain MIME information and can be left unset. For MIME specifications, see [RFC2045]. For the specification on mapping these properties, see [MS-OXCMAIL].

Type	Property	Content
PtypString	PidTagAttachMimeTag	The content-type MIME **header**.
PtypString	PidTagAttachContentId	A content **identifier** unique to this Message object that matches a corresponding "cid:" **Uniform Resource Identifier (URI)** scheme reference in the HTML body of the Message object.
PtypString	PidTagAttachContentLocation	A relative or full URI that matches a corresponding reference in the HTML body of the Message object.
PtypString	PidTagAttachContentBase	The base of a relative URI. MUST be set if PidTagAttachContentLocation contains a relative URI.

2.2.3 Message Object ROPS

The following sections specify the format of the ROP request buffers specific to the Message and Attachment Object Protocol. Before sending these requests to the server, the client has logged on to the server, and acquired a handle to the Message object or Folder object used in the **ROP request**.

2.2.3.1 RopOpenMessage Buffer Format

RopOpenMessage provides access to an existing Message object, which is identified by the **message ID (MID)**. The folder containing the Message object is identified by the **FID**.

For this ROP, the **InputHandleIndex** is either a **Store object** or a Folder Object. If a folder is used, it is not necessary that it is the parent folder, only that it is a folder within the same store. The **OutputHandleIndex** is a Message object.

When the server receives multiple requests to open the same Message object, it returns a different handle and maintains a separate transaction for each.

2.2.3.1.1 Request Buffer

The syntax of the RopOpenMessage request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopOpenMessage request buffer.

2.2.3.1.1.1 CodePageId

2-bytes specifying the code page in which the non-Unicode representation of the strings on this Message object are encoded. A value of 0x0FFF means that the code page of the **Logon object** is used.

2.2.3.1.1.2 FolderId

8-bytes containing the FID of the folder from which the message is to be opened.

2.2.3.1.1.3 OpenModeFlag

1-byte; MUST be one of the following.

Name	Value	Meaning
ReadOnly	0x00	Message will be opened as read only.
ReadWrite	0x01	Message will be opened for both reading and writing.
BestAccess	0x03	Open for read/write if possible, read-only if not.
OpenSoftDeleted	0x04	Open a **soft deleted** Message object if available.

2.2.3.1.1.4 MessageId

8-bytes containing the MID for the Message object to open.

2.2.3.1.2 Response Buffer

The syntax of the RopOpenMessage response buffer is specified in the [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopOpenMessage response buffer.

2.2.3.1.2.1 HasNamedProperties

1-byte.

Value	Meaning
0x00	No named properties are included in the ROP response buffer.
non-zero	named properties are included.

2.2.3.1.2.2 SubjectPrefix

A **TypedString** structure specifying the prefix for the subject of the Message object. Contains the value of the PidTagSubjectPrefix property.

2.2.3.1.2.3 NormalizedSubject

A **TypedString** structure specifying the normalized subject of the Message object. Contains the value of the PidTagNormalizedSubject property.

2.2.3.1.2.4 RecipientCount

A 2–BYTE unsigned **integer** containing the number of recipients associated with the Message object.

2.2.3.1.2.5 ColumnCount

A 2-BYTE unsigned **integer** containing the number of elements in the **RecipientColumns** field.

2.2.3.1.2.6 RecipientColumns

An array of **PropertyTag** structures with <ColumnCount> elements. Each element is valid for a recipient as specified in [MS-OXPROPS].

2.2.3.1.2.7 RowCount

A 1–BYTE unsigned **integer** containing the number of rows in the **RecipientRows** field. The value MUST be less than or equal to **RecipientCount**.

2.2.3.1.2.8 RecipientRows

An array of OpenRecipientRow structures with <RowCount> elements.

Element	Type	Size in bytes	Description
Recipient Type	Byte	1	See Recipient Type, below.
CodePageId	Integer	2	The code page in which the **non-Unicode** representation of the strings in the Recipient Data are encoded.
Reserved	Integer	2	MUST be "0x0000".
RecipientRowSize	Integer	2	The total number of bytes in the following **RecipientRow**.
RecipientRow	RecipientRow	RecipientRowSize	See [MS-OXCDATA].

A Recipient Type is a bitwise OR of one value from the Types table with zero or more values from the flags table.

Types are as follows.

Value	Description
0x01	**Primary recipient**.
0x02	**Carbon copy (Cc) recipient**.
0x03	**Bcc recipient**.

Flags are as follows.

Value	Description
0x10	When resending a previous failure this flag indicates that this recipient did not successfully receive the message on the previous attempt.
0x80	When resending a previous failure this flag indicates that this recipient did successfully receive the message on the previous attempt.

2.2.3.2 RopCreateMessage Buffer Format

RopCreateMessage is used to create a new Message object.

For this ROP, the **InputHandleIndex** is a folder or Logon object and the **OutputHandleIndex** is a Message object.

2.2.3.2.1 Request Buffer

The syntax of the RopCreateMessage request buffer is specified in the [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopCreateMessage request buffer.

2.2.3.2.1.1 CodePageId

2-bytes specifying the code page that the non-Unicode representation of the strings on this Message object are to be encoded; a value of "0x0FFF" means that the code page of the Logon object is used.

2.2.3.2.1.2 FolderId

8-bytes containing the FID for the Folder Object in which the Message object is to be created.

2.2.3.2.1.3 AssociatedFlag

1-byte.

Value	Meaning
0x00	Not an FAI message
non-zero	Is an FAI message

2.2.3.2.2 Response Buffer

The syntax of the RopCreateMessage response buffer is specified in the [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopCreateMessage response buffer.

2.2.3.2.2.1 HasMessageId

1-byte.

Value	Meaning
0x00	This is the last byte in the buffer.
non-zero	MessageId follows beginning with the next byte in the buffer.

2.2.3.2.2.2 MessageId

8-bytes containing the MID for the newly created Message object.

2.2.3.3 RopSaveChangesMessageBuffer Format

RopSaveChangesMessage commits the changes made to the Message object.

For this ROP, the **ResponseHandleIndex** is the containing Folder object or, for an embedded message, the **Embedded Message object**. The **InputHandleIndex** is a Message object.

When the server receives multiple requests to open the same Message object, it returns a different handle and maintains a separate transaction for each. Any changes made on one transaction MUST NOT be visible to another transaction until the changes are committed via RopSaveChangesMessage. Once a transaction on one handle has been committed, the server MUST return **ecObjectModified** for RopSaveChangesMessage requests on other handles and MUST NOT allow those transactions to be committed, unless the client instructs the server to override previous changes with the **ForceSave** flag from section 2.2.3.3.1.1.

2.2.3.3.1 Request Buffer

The syntax of the RopSaveChangesMessage request buffer is specified in the [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSaveChangesMessage request buffer.

2.2.3.3.1.1 SaveFlags

1-byte indicating the server save behavior; MUST be one value from the following table.

SaveType

Name	Value	Description
KeepOpenReadOnly	0x09	The client requests that the server commit the changes. The server either returns an error and leaves the Message object open with unchanged access level, or returns a success code and keeps the Message object open with read only access.
KeepOpenReadWrite	0x0A	The client requests that the server commit the changes. The server either returns an error and leaves the Message object open with unchanged access level, or returns a success code and keeps the Message object open with read/write access.
ForceSave	0x0C	The client requests that the server commit the changes. The server either returns an error and leaves the Message object open with unchanged access level, or returns a success code and keeps the Message object open with read/write access. The **ecObjectModified** error code is not valid when this flag is set: the server overwrites any changes instead.

2.2.3.3.2 Response Buffer

The syntax of the RopSaveChangesMessage response buffer is specified in the [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSaveChangesMessage response buffer.

2.2.3.3.2.1 Message Id

8-bytes containing the MID for the saved Message object.

2.2.3.4 RopRemoveAllRecipients Buffer Format

The client sends the RopRemoveAllRecipients request to delete all recipients from a message.

For this ROP the **InputHandleIndex** is a Message object.

2.2.3.4.1 Request Buffer

The syntax of the RopRemoveAllRecipients request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopRemoveAllRecipients request buffer.

2.2.3.4.1.1 Reserved

4-bytes; unspecified value.

2.2.3.4.2 Response Buffer

The syntax of the RopRemoveAllRecipients response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopRemoveAllRecipients response buffer.

2.2.3.5 RopModifyRecipients Buffer Format

RopModifyRecipients modifies recipients associated with the Message object.

For this ROP the **InputHandleIndex** is a Message object.

2.2.3.5.1 Request Buffer

The syntax of the RopModifyRecipients request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopModifyRecipients request buffer.

2.2.3.5.1.1 ColumnCount

2-bytes containing the number of elements in the **RecipientColumns** field. Is greater than or equal to "0x0000" and less than "0x7FEF".

2.2.3.5.1.2 RecipientColumns

An array of **PropertyTag** structures with <ColumnCount> elements. Each element is valid for a recipient as specified in [MS-OXPROPS]. The client MUST NOT include **property tags** for any properties that are part of an unextended RecipientRow, as specified in [MS-OXCDATA]:

PidTagAddressType PidTagDisplayName

PidTagEmailAddress PidTagEntryId

PidTagInstanceKey PidTagRecipientType

PidTagSearchKey PidTagSendRichInfo PidTagTransmittableDisplayName

2.2.3.5.1.3 RowCount

2-bytes containing the number of elements in the **RecipientRows** field. Is greater than or equal to "0x0000" and less than "0x7FEF".

2.2.3.5.1.4 RecipientRows

An array of ModifyRecipientRow structures with <RowCount> elements.

Element	Type	Size in bytes	Description
RowId	Integer	4	Row identifier.
RecipientType	Byte	1	See Recipient Type in section 2.2.3.1.2.8.
RecipientRowSize	Integer	2	The total number of bytes in the **RecipientRow** that follows.
RecipientRow	RecipientRow	RecipientRowSize	See [MS-OXCDATA].

2.2.3.5.2 Response Buffer

The syntax of the RopModifyRecipients response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopModifyRecipients response buffer.

2.2.3.6 RopReadRecipients Buffer Format

RopReadRecipients retrieves the recipients associated with the Message object.

For this ROP, the **InputHandleIndex** is a Message object.

2.2.3.6.1 Request Buffer

The syntax of the RopReadRecipients request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopReadRecipients request buffer.

2.2.3.6.1.1 RowId

4-bytes containing the starting index for the recipients to be retrieved.

2.2.3.6.1.2 Reserved

2-bytes; MUST be 0x0000.

2.2.3.6.2 Response Buffer

The syntax of the RopReadRecipients response buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopReadRecipients response buffer.

2.2.3.6.2.1 RowCount

2-bytes containing the number of elements in the **RecipientRows** field. Is greater than or equal " 0x0000" and less than "0x7FEF".

2.2.3.6.2.2 RecipientRows

An array of **ReadRecipientRow** structures with <RowCount> elements.

Element	Type	Size	Description
RowId	Integer	4	Index into the recipient list

Element	Type	Size	Description
Recipient Type	Byte	1	See Recipient Type in section 2.2.3.1.2.8
CodePageId	Integer	2	The code page that the non-Unicode representation of the strings in the **RecipientData** are encoded.
Reserved	n/a	2	MUST be "0x0000"
RecipientRowSize	Integer	2	The total number of bytes in the **RecipientRow**.
RecipientRow	RecipientRow	RecipientRowSize	See [MS-OXCDATA].

2.2.3.7 RopReloadCachedInformation Buffer Format

RopReloadCachedInformation retrieves the same information as RopOpenMessage but operates on an already opened Message object.

For this ROP the **InputHandleIndex** is a Message object.

2.2.3.7.1 Request Buffer

The syntax of the RopReloadCachedInformation request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopReloadCachedInformation request buffer.

2.2.3.7.1.1 Reserved

2-bytes; MUST be "0x0000".

2.2.3.7.2 Response Buffer

The response buffer for RopReloadCachedInformation is identical to the response buffer for RopOpenMessage.

2.2.3.8 RopSetMessageStatus Buffer Format

RopSetMessageStatus sets PidTagMessageStatus on a message in a folder without the need to open or save the Message object.

For this ROP the **InputHandleIndex** is a Folder object.

2.2.3.8.1 Request Buffer

The syntax of the RopSetMessageStatus request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSetMessageStatus request buffer.

2.2.3.8.1.1 MessageId

8-bytes containing the MID for the Message object to modify.

2.2.3.8.1.2 MessageStatusFlags

4-bytes containing the PidTagMessageStatus.

2.2.3.8.1.3 MessageStatusMask

4-bytes indicating which status flags are to be set and which are to be cleared. Contains a bitwise OR of zero or more values from the table in section 2.2.1.8. The server sets all flags that are set in both **MessageStatusMask** and **MessageStatusFlags**, and clears all flags that are set in **MessageStatusMask** but clear in **MessageStatusFlags**.

2.2.3.8.2 Response Buffer

The syntax of the RopSetMessageStatus response buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSetMessageStatus response buffer.

2.2.3.8.2.1 MessageStatusFlags

4-bytes indicating the status flags that were set on the Message object prior to processing this request. MUST contain a bitwise OR of zero or more values from the table in section 2.2.1.8.

2.2.3.9 RopGetMessageStatus Buffer Format

RopGetMessageStatus gets the message status of a message in a folder.

For this ROP the **InputHandleIndex** is a Folder object.

2.2.3.9.1 Request Buffer

The syntax of the RopGetMessageStatus request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopGetMessageStatus request buffer.

2.2.3.9.1.1 MessageId

8-bytes containing the MID for the Message object in which to operate.

2.2.3.9.2 Response Buffer

The syntax of the RopGetMessageStatus response buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSetMessageStatus response buffer.

2.2.3.9.2.1 MessageStatusFlags

4-bytes indicating the status of the Message object. Contains a bitwise OR of zero or more values from the table in section 2.2.1.8.

2.2.3.10 RopSetReadFlags Buffer Format

RopSetReadFlags changes the state of the PidTagMessageFlags property on one or more Message objects within a Folder object. It also triggers the sending of **read receipts**, as specified in [MS-OXOMSG].

For this ROP the **InputHandleIndex** is a Folder object.

2.2.3.10.1 Request Buffer

The syntax of the RopSetReadFlags request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSetReadFlags request buffer.

2.2.3.10.1.1 WantAsynchronous

1-byte indicating whether client is prepared for the RopSetReadFlags request to be processed asynchronously with status reported via RopProgress.

If the value is non-zero, the server SHOULD return a RopAsyncProgress response, but MAY return a RopSetReadFlags response instead. For more details on the asynchronous behavior of ROPs see [MS-OXCROPS].

2.2.3.10.1.2 ReadFlags

1-byte containing a bitwise OR of zero or more values from the following table. The server modifies bits on the PidTagMessageFlags property. The flags, **rfGenerateReceiptOnly**, **rfsuppressReceipt**, and **rfClearReadFlag**, (**rfClearNotifyRead** OR **rfClearNotifyUnread**), are mutually exclusive.

Name	Value	Description
rfDefault	0x00	The server sets the read flag and sends the receipt.
rfSuppressReceipt	0x01	The user requests that any pending read report be canceled; Server sets **mfRead** bit.
rfReserved	0x0A	Ignored by the server.
rfClearReadFlag	0x04	Server clears the mfRead bit; Client MUST include **rfSuppressReceipt** with this flag.
rfGenerateReceiptOnly	0x10	The server sends a read report if one is pending, but does not change the **mfRead** bit.
rfClearNotifyRead	0x20	The server clears the **mfNotifyRead** bit, but does not send a read report.
rfClearNotifyUnread	0x40	The server clears the **mfNotifyUnread** bit, but does not send a non-read report.

2.2.3.10.1.3 MessageIdCount

2-bytes containing the number of elements in the **MessageIds** field.

2.2.3.10.1.4 MessageIds

An array of MIDs with <MessageIdCount> elements.

2.2.3.10.2 Response Buffer

The syntax of the RopSetReadFlags response buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSetReadFlags response buffer.

2.2.3.10.2.1 PartialCompletion

1-byte indicating the server was unable to modify one or more of the Message objects represented in the **MessageIds** field.

2.2.3.11 RopSetMessageReadFlag

RopSetMessageReadFlag changes the state of the PidTagMessageFlags property for the Message object. It also triggers the sending of read receipts, as specified in [MS-OXOMSG].

In this section, "in **public folder** mode" means that the logon associated with the **LogonID** from the request was created with the Private flag unset ([MS-OXCSTOR]).

For this ROP the ResponseHandleIndex is a Folder object and the InputHandleIndex is a Message object.

2.2.3.11.1 Request Buffer

The syntax of the RopSetMessageReadFlag request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSetMessageReadFlag request buffer.

2.2.3.11.1.1 ReadFlags

1-byte containing a bitwise OR of one or more values from the table in section 2.2.3.10.1.2

2.2.3.11.1.2 ClientData

24-bytes containing a **LongTermID** (see [MS-OXCDATA]) for the user that is making the ROP request when in public folder mode; 0-bytes otherwise.

2.2.3.11.2 Response Buffer

The syntax of the RopSetMessageReadFlag response buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSetMessageReadFlag response buffer.

2.2.3.11.2.1 ReadStatusChanged

1-byte:

Value	Description
0x00	The read status on the Message object was unchanged or the logon is not in public folder mode.
non-zero	The read status on the Message object changed and the logon is in public folder mode.

2.2.3.11.2.2 LogonId

1-byte; containing the LogonID from the request when the value in **ReadStatusChanged** is non-zero; 0-bytes otherwise.

2.2.3.11.2.3 ClientData

24-bytes; containing the **ClientData** from the request when the value in **ReadStatusChanged** is non-zero; 0-bytes otherwise.

2.2.3.12 RopOpenAttachment

RopOpenAttachment opens an Attachment object stored on the Message object.

For this ROP the **InputHandleIndex** is a Message object and the **OutputHandleIndex** is an Attachment object.

2.2.3.12.1 Request Buffer

The syntax of the RopOpenAttachment request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopOpenAttachment request buffer.

2.2.3.12.1.1 OpenAttachmentFlags

1-byte containing one of the following values.

Name	Value	Meaning
ReadOnly	0x00	Message will be opened as read only.
ReadWrite	0x01	Message will be opened for both reading and writing.
BestAccess	0x03	Open for read/write if possible, read-only if not.

2.2.3.12.1.2 AttachmentID

4-bytes containing the ID of the Attachment object to be opened. See PidTagAttachNumber.

2.2.3.12.2 Response Buffer

The syntax of the RopOpenAttachment response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopOpenAttachment response buffer.

2.2.3.13 RopCreateAttachment

RopCreateAttachment creates a new Attachment object on the Message object.

For this ROP the **InputHandleIndex** is a Message object and the **OutputHandleIndex** is an Attachment object.

2.2.3.13.1 Request Buffer

The syntax of the RopCreateAttachment request buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopCreateAttachment request buffer.

2.2.3.13.2 Response Buffer

The syntax of the RopCreateAttachment response buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopCreateAttachment response buffer.

2.2.3.13.2.1 AttachmentID

4-bytes containing the ID for the Attachment object that was created. See PidTagAttachNumber.

2.2.3.14 RopDeleteAttachment

RopDeleteAttachment deletes an existing Attachment object from the Message object.

For this ROP, the **InputHandleIndex** is a Message object.

2.2.3.14.1 Request Buffer

The syntax of the RopDeleteAttachment request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopDeleteAttachment request buffer.

2.2.3.14.1.1 AttachmentID

4-bytes containing the ID of the Attachment object to be deleted. See PidTagAttachNumber.

2.2.3.14.2 Response Buffer

The syntax of the RopDeleteAttachment response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopDeleteAttachment response buffer.

2.2.3.15 RopSaveChangesAttachment Buffer Format

RopSaveChangesAttachment commits the changes made to the Attachment object.

For this ROP, the **ResponseHandleIndex** is the containing Message object and **InputHandleIndex** is an Attachment object.

2.2.3.15.1 Request Buffer

The syntax of the RopSaveChangesAttachment request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopSaveChangesAttachment request buffer.

2.2.3.15.1.1 SaveFlags

See section 2.2.3.3.1.1.

2.2.3.15.2 Response Buffer

The syntax of the **RopSaveChangesAttachment** response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the **RopSaveChangesAttachment** response buffer.

2.2.3.16 RopOpenEmbeddedMessage Buffer Format

RopOpenEmbeddedMessage retrieves a handle to a Message object from the given Attachment object.

For this ROP the **InputHandleIndex** is an Attachment object and the **OutputHandleIndex** is a Message object.

2.2.3.16.1 Request Buffer

The syntax of the RopOpenEmbeddedMessage request buffer is specified in [MS-OXCROPS]. The fields specified in the following sub-sections are part of the RopOpenEmbeddedMessage request buffer.

2.2.3.16.1.1 CodePageId

2-bytes specifying the code page in which the non-Unicode representation of the strings on this Message object MUST be encoded.

2.2.3.16.1.2 OpenModeFlags

1-byte.

Name	Value	Meaning
ReadOnly	0x00	Message will be opened as read only.
ReadWrite	0x01	Message will be opened for both reading and writing.
Create	0x02	Create the attachment if it does not already exist and open the message for both reading and writing.

2.2.3.16.2 Response Buffer

The syntax of the RopOpenEmbeddedMessage response buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopOpenEmbeddedMessage response buffer.

2.2.3.16.2.1 MessageId

8 bytes containing the MID for the Message object.

2.2.3.16.2.2 HasNamedProperties

See section 2.2.3.1.2.1

2.2.3.16.2.3 Subject Prefix

See section 2.2.3.1.2.2.

2.2.3.16.2.4 NormalizedSubject

See section 2.2.3.1.2.3.

2.2.3.16.2.5 RecipientCount

See section 2.2.3.1.2.4.

2.2.3.16.2.6 ColumnCount

See section 2.2.3.1.2.5.

2.2.3.16.2.7 RecipientColumns

See section 2.2.3.1.2.6.

2.2.3.16.2.8 RowCount

See section 2.2.3.1.2.7.

2.2.3.16.2.9 RecipientRows

See section 2.2.3.1.2.8.

2.2.3.17 RopGetAttachmentTable Buffer Format

RopGetAttachmentTable retrieves a handle to a table object that represents the attachments stored on the Message object. See [MS-OXCTABL] for more details on table objects.

For this ROP the **InputHandleIndex** is a Message object and the **OutputHandleIndex** is a table object.

2.2.3.17.1 Request Buffer

The syntax of the RopGetAttachmentTable request buffer is specified in [MS-OXCROPS].

The fields specified in the following sub-sections are part of the RopGetAttachmentTable request buffer.

2.2.3.17.1.1 Table Flags

1-byte.

Name	Value	Description
Standard	0x00	Open the table.
Unicode	0x40	Open the table. Also requests that the columns containing string data be returned in Unicode format.

2.2.3.17.2 Response Buffer

The syntax of the RopGetAttachmentTable response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopGetAttachmentTable response buffer.

2.2.3.18 RopGetValidAttachments Buffer Format

RopGetValidAttachments gets the valid attachment identifiers of a **message**.

2.2.3.18.1 Request Buffer

The syntax of the RopGetValidAttachments request buffer is specified in [MS-OXCROPS].

2.2.3.18.2 Success Response Buffer

The syntax of the RopGetValidAttachments success response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopGetValidAttachments success response buffer.

2.2.3.18.3 Failure Response Buffer

The syntax of the RopGetValidAttachments failure response buffer is specified in [MS-OXCROPS].

This protocol adds no additional field information to the RopGetValidAttachments failure response buffer.

3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

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

3.1.1.1 Linking a Contact Object

To link a Contact object with another Message object, the client sets the following properties. See section 2.2.1.20 for more details.

- PidLidContactLinkEntry
- PidLidContactLinkName
- PidLidContactLinkSearchKey
- PidLidContacts

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Opening a Message Object

When a higher layer needs to obtain a handle to an existing Message object the client sends a RopOpenMessage request. In order to send this request, the client first obtains the MID for the Message object to be opened, and either the FID or the LogonID. The MID is accessible from the contents table of the Folder Object that contains the Message object by including PidTagMid in a RopSetColumns request. For more details, see [MS-OXCTABL].

To open a soft deleted Message object, the client MUST include **OpenSoftDeleted** in the **OpenModeFlag** field.

When the client receives the response buffer, it caches the data from the **NormalizedSubject** and **SubjectPrefix** fields; it also updates the cache when issuing RopSetProperties for PidTagNormalizedSubject and PidTagSubjectPrefix and uses the cached values.

The client uses the opened Message object in subsequent ROPs; it MUST eventually send a RopRelease request on the Message object, and after doing so, MUST NOT use the Message object for any subsequent ROPs.

The client is responsible for maintaining the privacy of the properties on the Message object when PidLidPrivate is set to "0x01".

3.1.4.2 Creating a Message Object

When a higher layer needs to create a new Message object, the client sends a RopCreateMessage request. The client sends a RopSaveChangesMessage request to commit the new Message object, and uses the opened Message object in subsequent ROPs. It MUST eventually send a RopRelease request on the Message object and after doing so MUST NOT use the Message object for any subsequent ROPs.

3.1.4.3 Saving Message Object Changes

When a higher layer wants to save all the changes to a Message object, the client sends a RopSaveChangesMessage request.

The client controls the access level of the Message object handle after saving changes by setting the proper flags as specified in section 2.2.3.3.1.1.

3.1.4.4 Removing All Recipients

When a higher layer wants to clear all recipients from a Message object, the client sends a RopRemoveAllRecipients request.

The client sends a RopSaveChangesMessage for the Message object associated with the removed recipients in order to commit the changes.

3.1.4.5 Adding, Deleting, and Modifying Recipients

When a higher layer wants to modify recipients of the Message object, the client sends a RopModifyRecipients request.

To modify an existing recipient the client sets the **RowId** field to the **RowId** of the recipient to be modified and sets all of the **ModifyRecipientRow** data to the desired values for that recipient, including any additional property information for the recipients. Additional property information is set by adding PropertyTag values to the **RecipientColumns** field and including the property values in the **RecipientProperties** field.

To delete an existing recipient the client sets the **RowId** field to the **RowId** of the recipient to be deleted and sets the **RecipientRowSize** field to "0x0000".

To add a new recipient the client sets the **RowId** field to a value greater than the largest **RowId** for any recipient that already exists on the Message object. The client sets all of the **ModifyRecipientRow** data to the desired values for that recipient, including any additional property information.

The client sends a RopSaveChangesMessage for the Message object associated with the added recipients in order to commit the changes.

3.1.4.6 Reading Recipients

When a higher layer want a list of all recipients on the Message object, the client sends a RopReadRecipients request.

If the COUNT of recipients and the COUNT of recipient rows in the RopOpenMessage response buffer are the same, then the client uses the **RecipientRow** information from RopOpenMessage instead of

sending a RopReadRecipients request. If the counts are not equal then the response buffers for a series of RopReadRecipients contain all the recipients associated with the Message object including those returned in the RopOpenMessage response buffer.

A client accesses the information for all recipients in the message by setting the **RowId** field to "0x0000", and then iteratively sending RopReadRecipients with an increasing **RowId** value to obtain the recipients that did not fit in the previous request.

3.1.4.7 Reload Message Object Header Info

When a higher layer wants to retrieve the current state of the data returned in RopOpenMessage then the client sends a RopReloadCachedInformation request.

3.1.4.8 Setting Message Status

When a higher layer is working with a header message object and wants to change its status, (mark or unmark the header message object for download or delete), the client sends a RopSetMessageStatus request.

To modify the status of a header message object:

1. Obtain the message's MID, see section 3.1.4.1.
2. Send the RopSetMessageStatus request, setting the mask and status appropriately.

3.1.4.9 Getting Message Status

When a higher layer is working with a header message object and wants to check its status, the client sends a RopGetMessageStatus request.

To retrieve the status of a header message object:

1. Obtain the message's MID, see section 3.1.4.1.
2. Sent the RopGetMessageStatus request; if the request succeeds then the header message object's PidTagMessageStatus value is returned in the response buffer.

3.1.4.10 Setting Message Object Read State

When a higher layer wants to mark one or more Message objects as read or unread without opening the Message objects, the client sends a RopSetReadFlags request. The client obtains a list of MIDs using a contents table; see section 3.1.4.1, and uses the list of MIDs in the RopSetReadFlags request.

When a higher layer wants to mark or unmark a single opened Message object as read, the client sends a RopSetMessageReadFlag request.

The client controls whether the Message object is marked as read or unread, as well as the sending of read receipts by setting the appropriate flags as specified in section 2.2.3.10.1.2.

3.1.4.11 Opening Attachments

When a higher layer wants to open and manipulate an existing Attachment object to a Message object, the client sends a RopOpenAttachment request.

The client MUST use a valid AttachmentID (see section 2.2.3.12.1.2) when requesting to open an attachment. See section 3.1.4.16.

The client uses the opened Attachment object in subsequent ROPs. It eventually sends a RopRelease request on the Attachment object and after doing so, MUST NOT use the Attachment object for any subsequent ROPs.

3.1.4.12 Creating Attachments

When a higher layer wants to add a new Attachment object to a Message object, the client sends a RopCreateAttachment request.

The client sends a RopSaveChangesAttachment request to commit the new Attachment object, and uses the newly created Attachment object in subsequent ROPs. The client eventually sends a RopRelease request on the Attachment object and after doing so, MUST NOT use the Attachment object for any subsequent ROPs.

The client sends a RopSaveChangesMessage request to commit the Attachment object change to the Message object.

3.1.4.13 Setting Attachment Object Content

When a higher layer wants to add the contents of a file to an Attachment object, the client sends a RopSetProperties request as specified in [MS-OXCPRPT].

Depending on the type of Attachment object the higher layer intends to use, the client sets the appropriate value for PidTagAttachMethod as specified in section 2.2.2.9.

The client sends a RopSaveChangesAttachment request to commit the change to the Attachment object and a RopSaveChangesMessage request to commit the Attachment object change to the Message object.

3.1.4.14 Saving Attachment Object Changes

When a higher layer wants to save changes to an Attachment object, the client sends a RopSaveChangesAttachment request. It sends a RopSaveChangesMessage request to commit the Attachment object changes to the Message object.

3.1.4.15 Opening an Embedded Message Object

When a higher layer wants to open an existing Attachment object and access and manipulate it as if it were a Message object, the client sends a RopOpenEmbeddedMessage request.

The client uses the opened Message object in subsequent ROPs; it eventually sends a RopRelease request on the Message object and after doing so, MUST NOT use the Message object for any subsequent ROPs.

3.1.4.16 Accessing the Attachments Table

When a higher layer wants to retrieve information about all Attachment objects associated with a Message object without opening each Attachment object, the client sends a RopGetAttachmentTable request.

The server returns a table of properties for each Attachment object associated with the Message object as specified in [MS-OXCTABL]. To retrieve the AttachmentID, the client includes PidTagAttachNumber when sending a RopSetColumns request.

3.1.4.17 Creating an Embedded Message

When a higher layer wants to create an embedded message, the client sends a RopCreateAttachment to create an attachment on a message. The client uses RopSetProperties to set the value of PidTagAttachMethod to **afEmbeddedMessage**. Finally the client sends a RopOpenEmbeddedMessage on the attachment to get a Message object handle.

3.1.4.18 Saving an Embedded Message

When a higher layer wants to save an embedded message, the client sends a RopSaveChangesMessage on the embedded message. Then the client sends a RopSaveChangesAttachment on the attachment from which the embedded message was opened. Finally, the client sends a RopSaveChangesMessage on the enclosing message.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 RopOpenMessage

Provides access to existing Message objects stored by the server. The Message object returned by this ROP is used in subsequent ROPs, such as RopGetPropertiesSpecific. See specific ROPs for information of which operate on Message objects, as specified in [MS-OXCROPS].

RopOpenMessage MUST NOT succeed if a Message object with the specified ID does not exist or the client has insufficient access rights to view the Message object.

If the **OpenModeFlag** field includes the **OpenSoftDeleted** flag, then RopOpenMessage provides access to all Message objects, including soft deleted Message objects. If **OpenSoftDeleted** is not included, then the server MUST NOT provide access to soft deleted Message objects.

The response field **RecipientCount** indicates the current number of recipients in the message. In addition, the server returns data for as many recipients as will fit in the response buffer, in order of **RowId** value. The data for each recipient is encoded as an **OpenRecipientRow** structure in the **RecipientRows** field. The response field **RowCount** indicates how many recipients are present in **RecipientRows**.

The following are specific error codes which apply to this ROP.

Name	Value	Meaning
ecNotFound	0x8004010F	The MID does not correspond to a message in the database. The user does not have rights to the message. The message is soft deleted and the client has not specified the **OpenSoftDeleted** flag as part of the **OpenModeFlag** field.
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a folder or Logon object.

3.2.5.2 RopCreateMessage

Creates a new Message object on the server and provides access to it by returning a Message object handle for use in subsequent ROPs. The server MUST NOT commit the new Message object until it receives a RopSaveChangesMessage request.

The server MUST initialize the following properties before responding.

Property name	Initial Data
PidTagImportance	"0x00000001"
PidTagMessageClass	"IPM.Note"
PidTagSensitivity	"0x00000000"
PidTagDisplayBcc	""
PidTagDisplayCc	""
PidTagDisplayTo	""
PidTagMessageFlags	"0x00000009"
PidTagMessageSize	Calculated based on the data
PidTagHasAttachments	"0x00"
PidTagSecurityDescriptor	See PidTagSecurityDescriptor
PidTagUrlCompNameSet	"0x00"
PidTagTrustSender	"0x00000001"
PidTagAccess	"0x00000003"

Property name	Initial Data
PidTagAccessLevel	"0x00000001"
PidTagUrlCompName	"No Subject.EML"
PidTagCreationTime	Time RopCreateMessage was processed
PidTagLastModificationTime	Same as PidTagCreationTime
PidTagSearchKey	Server generated SearchKey.
PidTagMessageLocaleId	The Logon object LocaleID.
PidTagCreatorName	Name of the creator.
PidTagCreatorEntryId	Address Book EntryID of the creator
PidTagLastModifierName	Same as PidTagCreatorName
PidTagLastModifierEntryId	Same as PidTagCreatorEntryId
PidTagHasNamedProperties	"0x00"
PidTagLocaleId	Same as PidTagMessageLocaleId
PidTagLocalCommitTime	Same as PidTagCreationTime

The following specific error code applies to this ROP.

Name	Value	Meaning
ecAccessDenied	0x80000009	The user does not have **permissions** to create this message.

3.2.5.3 RopSaveChangesMessage

Commits the changes made to a Message object on the server. The status of the Message object after the commit is determined by the value of the **SaveFlags** as documented in section 2.2.3.3.1.1.

The response contains the message ID of the committed message.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The values of the **SaveFlags** are not a supported combination as documented in section 2.2.3.3.1.1.
ecObjectModified	0x80040109	The underlying data for this Message object was changed through another transaction context.

3.2.5.4 RopRemoveAllRecipients

Removes all the recipients from a Message object.

The server ignores the value of Reserved.

Until the server receives a RopSaveChangesMessage request from the client, the server adheres to the following:

- The **RowIds** and associated data of removed recipients MUST NOT be returned as part of any subsequent handling of ROPs for the opened Message object.
- The changes made to the recipients MUST NOT be included in the response buffer returned for ROP requests that apply To recipients on Message object handles.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a Message object.

3.2.5.5 RopModifyRecipients

Modifies the recipients on a Message object according to the data in the request buffer. The format of the request buffer is specified in section 2.2.3.5.1.

For each recipient provided the server locates its representation of the recipient based on **RowId**. If the recipient indicated by **RowId** does not exist then the server creates a new recipient with that **RowId** and applies the data from the request.

If the recipient currently exists on the Message object and the value of **RecipientRowSize** in the request buffer is non-zero, the server replaces all existing properties of the recipient with the property values supplied in the request. If the value of **RecipientRowSize** in the request buffer is "0x0000" then the server deletes the recipient from the Message object.

Until the server receives a RopSaveChangesMessage request from the client, the server adheres to the following:

- If a recipient was deleted, its **RowId** and associated data MUST NOT be returned as part of any subsequent handling of ROPs for the opened Message object.
- Any changes made to the recipients MUST be included in the response buffer for any subsequent ROP requests that apply To recipients for the same Message object handle.
- The changes made to the recipients MUST NOT be included in the response buffer returned for ROP requests that apply To recipients on different Message object handles.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a Message object.

3.2.5.6 RopReadRecipients

Provides the recipient information for the Message object in a tabular form, where each row has information for a single recipient. RopReadRecipients is used to obtain information for all recipients in the Message object, regardless of the number of recipients on the message.

The server provides the recipient information starting with the recipient specified by the **RowId** field. If there is a recipient with the given **RowId**, the server provides the information for that recipient and as many recipients as possible, limited by the number of actual recipients in the message and the amount of recipient information that fits in the response buffer.

When **RowId** is "0x0000", the server chooses the first recipient in the message even if its **RowId** does not match. If the Message does not have recipients, the server returns **ecNotFound**.

The following specific error codes apply to this ROP.

Name	Value	Meaning
ecNotFound	0x8004010F	Recipient row **RowId** does not exist on the message.
ecBufferTooSmall	0x0000047D	Unable to fit at least one recipient in the response buffer. See [MS-OXCROPS] for specific handling.
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a Message object.

3.2.5.7 RopReloadCachedInformation

Returns the same information as would be returned by RopOpenMessage, except that it is updated with the information that has been modified on the Message object.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a Message object.

3.2.5.8 RopSetMessageStatus

Modifies the PidTagMessageStatus property of a single message.

The server modifies the bits on this property specified by the **MessageStatusMask**.

The server immediately commits the changes to the Message object as if the Message object had been opened and RopSaveChangesMessage had been called, except that it only changes PidTagMessageStatus, not PidTagChangeKey, PidTagLastModificationTime, or any other property that is modified during RopSaveChangesMessage.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a Folder object.

3.2.5.9 RopGetMessageStatus

Provides the value of PidTagMessageStatus property of a single message; MUST NOT require the Message object to be opened.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a Folder object.

3.2.5.10 RopSetReadFlags

Modifies the PidTagMessageFlags property of several messages.

The server immediately commits the changes to the Message objects as if the Message objects had been opened and RopSaveMessageChanges had been called, except that it only changes PidTagMessageFlags, not PidTagChangeKey, PidTagLastModificationTime, or any other property that is modified during RopSaveChangesMessage.

If the **WantAsynchronous** flag is non-zero, then the server MAY execute this ROP asynchronously. For more details, see RopProgress in [MS-OXCPRPT].

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The InputHandleIndex on which this ROP was called does not refer to a Folder object.

3.2.5.11 RopSetMessageReadFlag

Modifies the PidTagMessageFlags property of a single message.

The server immediately commits the changes to the Message object as if the Message object had been opened and RopSaveChangesMessage had been called, except that it only changes PidTagMessageFlags, not PidTagChangeKey, PidTagLastModificationTime, or any other property that is modified during RopSaveChangesMessage.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The InputHandleIndex on which this ROP was called does not refer to a Message object.

3.2.5.12 RopOpenAttachment

Provides access to a single existing Attachment object. The handle returned by this ROP is used in subsequent ROPs, such as RopGetPropertiesSpecific. See specific ROPs for information of which operate on Attachment objects [MS-OXCROPS].

The following specific error codes apply to this ROP.

Name	Value	Meaning
ecNotFound	0x8004010F	The AttachmentID does not correspond to an attachment on the Message object.

Name	Value	Meaning
ecAccessDenied	0x80000009	The user has insufficient privileges.
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to a Message object.

3.2.5.13 RopCreateAttachment

Creates a new Attachment object and provides a handle to it for use in subsequent ROPs. The server does not commit the new Attachment object until it receives a call to RopSaveChangesAttachment.

The server MUST initialize the following properties before responding.

PropertyName	Initial Data
PidTagAttachNumber	Varies, depending on the number of existing attachments on the Message object.
PidTagAttachSize	"0x00000000 "
PidTagAccessLevel	"0x00000001"
PidTagRenderingPosition	"0x00000000"
PidTagCreationTime	Time RopCreateAttachment was processed.
PidTagLastModificationTime	Same as PidTagCreationTime

The following specific error codes apply to this ROP.

Name	Value	Meaning
ecAccessDenied	0x80000009	The user does not have permissions to create an attachment on this message.
ecMaxAttachmentExceeded	0x000004DB	The (server defined) maximum number of attachments for a message has been exceeded.
ecNotSupported	0x80040102	The InputHandleIndex on which this ROP was called does not refer to a Message object.

3.2.5.14 RopSaveChangesAttachment

Commits the changes made to an Attachment object. The status of the Attachment object after the commit is determined by the values of the **SaveFlags** as documented in section 2.2.3.3.1.1.

Although the server commits any pending changes to the Attachment object in the context of its containing Message object, the changes MUST NOT be committed to the database until RopSaveChangesMessage has been executed on the handle of the Message object.

The following specific error code applies to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The value of **SaveFlags** is not a supported combination as documented

Name	Value	Meaning
		in section 2.2.3.3.1.1. The **InputHandleIndex** on which this ROP was called does not refer to an Attachment object.

3.2.5.15 RopDeleteAttachment

Removes an Attachment object from a message. The server recalculates PidTagHasAttachments during this ROP.

The following specific error codes apply to this ROP.

Name	Value	Meaning
ecNotFound	0x8004010F	The AttachmentID does not correspond to an attachment on the Message object.
ecAccessDenied	0x80000009	The user has insufficient privileges.
ecNotSupported	0x80040102	The InputHandleIndex on which this ROP was called does not refer to a Message object.

3.2.5.16 RopOpenEmbeddedMessage

Provides access to an Embedded Message object stored in an Attachment object. If the embedded object does not exist, then the client creates an Attachment object following the process defined in section 3.1.4.17. Once the attachment is created and its properties initialized, as specified in section 3.1.4.17, the client sends a RopOpenEmbeddedMessage on the attachment to get a Message object handle. The returned handle is used in subsequent ROPs (similar to the one returned by RopOpenMessage). The server MUST NOT commit the Message object to the containing Attachment object until RopSaveChangesMessage is called with the Embedded Message object's handle.

The following specific error codes apply to this ROP.

Name	Value	Meaning
ecAccessDenied	0x80000009	The user does not have permission to open or create this message.
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to an Attachment object.
ecUnknownCodePage	0x000003ef	The code page is unknown

3.2.5.17 RopGetAttachmentTable

Returns a handle to a table object for use in subsequent ROPs as specified in [MS-OXCTABL]. The table object returned allows access to the properties of Attachment objects.

The following specific error codes apply to this ROP.

Name	Value	Meaning
ecNotSupported	0x80040102	The **InputHandleIndex** on which this ROP was called does not refer to

Name	Value	Meaning
		a Message object.
ecBusy	0x80040108	The server is too busy to complete the request.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

A user creates a new HTML-format e-mail, sets its subject to "abc123sample" and its body to "This is a sample body text". The user also adds two attachments: an HTML embedded image and a text file, adds a recipient, then saves and closes the message.

4.1 Create Message

The client first creates a new Message object by sending a RopCreateMessage request.

4.1.1 RopCreateMessage Request Buffer

```
0000: 06 00 00 01 ff 0f 01 00-00 00 00 f0 79 93 00
```

RopId: 0x06

LogonId: 0x00

InputHandleIndex: 0x00

OutputHandleIndex: 0x01

CodePageId: 0x0FFF

FolderId:01 00 00 00 00 f0 79 93

AssociatedFlag: 0x00

4.1.2 RopCreateMessage Response Buffer

```
0000: 06 01 00 00 00 00 00
```

RopId: 0x06

OutputHandleIndex: 0x01

ReturnValue: 0x00000000

HasMessageId: 0x00

4.2 Name to Id Mapping

Before manipulating named properties on Message objects, the client needs to ask the server to perform a mapping from the named properties to **property identifiers**, using RopGetPropertyIdsFromNames as specified in [MS-OXCPRPT].

4.3 Get Attachment Table

The client sends a RopGetAttachmentTable request to retrieve the attachment table for a Message object.

4.3.1 RopGetAttachmentTable Request Buffer

```
0000:21 00 00 01 00
```

RopId: 0x21

LogonId: 0x00

InputHandleIndex: 0x00

OutputHandleIndex: 0x01

TableFlags: 0x00 (Standard)

4.3.2 RopGetAttachmentTable Response Buffer

```
0000:21 01 00 00 00 00
```

RopId: 0x21

OutputHandleIndex: 0x01

ReturnValue: 0x00000000

4.4 Insert HTML Embedded Image

The client first creates the Attachment object on the Message object, then sets its properties and commits the changes.

4.4.1 RopCreateAttachment Request Buffer

```
0000: 23 00 00 01
```

RopId: 0x23

LogonId: 0x00

InputHandleIndex: 0x00

OutputHandleIndex: 0x01

4.4.2 RopCreateAttachment Response Buffer

```
0000: 23 01 00 00 00 00 00 00-00 00
```

RopId: 0x23

OutputHandleIndex: 0x01

ReturnValue: 0x00000000

AttachmentID: 0x00000000

4.4.3 Setting Properties

At this point the client uses RopSetProperties as specified in [MS-OXCPRPT] to set properties on the Attachment objects.

Property tag	Property name	Data
0x37050003	PidTagAttachMethod	"0x00000001"
0x370b0003	PidTagRenderingPosition	"0xFFFFFFFF"
0x7ffd0003	PidTagAttachmentFlags	"0x00000000"
0x3001001f	PidTagDisplayName	"image001.PNG"
0x3712001f	PidTagAttachContentId	"image001.PNG@01C86E1C.F1954390"
0x370e001f	PidTagAttachMimeTag	"image/PNG"
0x7ffa0003	PidTagAttachmentLinkId	"0x00000000"
0x37140003	PidTagAttachFlags	"0x00000004"
0x7ffe000b	PidTagAttachmentHidden	"0x01"
0x3707001f	PidTagAttachLongFilename	"image001.PNG"
0x3704001f	PidTagAttachFilename	"image001.PNG"
0x3703001f	PidTagAttachExtension	".PNG"

To set the contents of the embedded image, the client uses four ROPs, as described in [MS-OXCPRPT].

RopOpenStream with PidTagAttachDataBinary.

RopSetStreamSize with the size of image file data.

RopWriteStream request with the actual file contents.

RopRelease for the handle returned from RopOpenStream.

4.4.4 RopSaveChangesAttachment Request Buffer

```
0000: 25 00 01 00 02
```

RopId: 0x25

LogonId:0x00

ResponseHandleIndex: 0x01

InputHandleIndex: 0x00

SaveFlags: 0x0A (**KeepOpenReadWrite**)

4.4.5 RopSaveChangesAttachment Response Buffer

```
0000: 25 01 00 00 00 00
```

RopId: 0x25

ResponseHandleIndex: 0x01

ReturnValue: 0x00000000

4.4.6 Releasing Attachment Object

Finally, the client releases the Attachment object by using RopRelease, as specified in [MS-OXCROPS].

4.5 Attach Text File

The client first creates the Attachment object on the Message object, then sets its properties and commits the changes.

4.5.1 RopCreateAttachment Request Buffer

```
0000:23 00 00 03
```

RopId: 0x23

LogonId: 0x00

InputHandleIndex: 0x00

OutputHandleIndex: 0x03

4.5.2 RopCreateAttachment Response Buffer

```
0000: 23 03 00 00 00 00 01 00-00 00
```

RopId: 0x23

OutputHandleIndex: 0x03

ReturnValue: 0x00000000

AttachmentID: 0x00000001

4.5.3 Setting Properties

At this point the client uses RopSetProperties as specified in [MS-OXCPRPT] to set properties on the Attachment objects.

Property tag	Property name	Data
0x37050003	PidTagAttachMethod	"0x00000001"

Property tag	Property name	Data
0x370b0003	PidTagRenderingPosition	"0xFFFFFFFF"
0x7ffd0003	PidTagAttachmentFlags	"0x00000000"
0x3001001f	PidTagDisplayName	"test.txt"
0x7ffa0003	PidTagAttachmentLinkId	"0x00000000"
0x37140003	PidTagAttachFlags	"0x00000000"
0x7ffe000b	PidTagAttachmentHidden	"0x00"
0x3707001f	PidTagAttachLongFilename	"test.txt"
0x3704001f	PidTagAttachFilename	"test.txt"
0x3703001f	PidTagAttachExtension	".txt"
0x30070040	PidTagCreationTime	"2008/02/1222:28:34.636"
0x30080040	PidTagLastModificationTime	"2008/02/1222:28:50.112"
0x37090102	PidTagAttachRendering	3512 Bytes of Windows Metafile

To set the contents of the embedded image, the client uses four ROPs. For more information, see [MS-OXCPRPT].

1. RopOpenStream with PidTagAttachDataBinary.
2. RopSetStreamSize with the size of image file data.
3. RopWriteStream request with the actual file contents.
4. RopRelease for the handle returned from RopOpenStream.

4.5.4 RopSaveChangesAttachment Request Buffer

```
0000: 25 00 02 01 02
```

RopId: 0x25

LogonId: 0x00

ResponseHandleIndex: 0x02

InputHandleIndex: 0x01

SaveFlags: 0x0A (**KeepOpenReadWrite**)

4.5.5 RopSaveChangesAttachment Response Buffer

```
0000: 25 02 00 00 00 00
```

RopId: 0x25

ResponseHandleIndex: 0x02

ReturnValue: 0x00000000

4.5.6 Releasing Attachment Object

Finally, the client releases the Attachment object by using RopRelease, as specified in [MS-OXCROPS].

4.6 Setting Message Properties

The client sets all the necessary properties using RopSetProperties as specified in [MS-OXCPRPT].

The HTML body, stored in PidTagBodyHtml, is the following:

```
<html>
<head>
<meta http-equiv=Content-Type content="text/html; charset=us-ascii">
</head>
<body lang=EN-US link=blue vlink=purple>
<p>This is a sample body text<o:p></o:p></p>
<p ><img width=174 height=152 id="Picture_x0020_2"
src="cid:image001.png@01C86E1C.F1954390"
alt="cid:image001.png@01C86E1C.F1954390"><o:p></o:p></p>
</div>
</body>
</html>
```

4.7 Adding Recipients

4.7.1 RopModifyRecipients Request Buffer

```
0000:0e 00 08 0c 00 03 00 fe-0f 03 00 00 39 1f 00 ff
0010:39 1f 00 fe 39 03 00 71-3a 03 00 05 39 1f 00 f6
0020:5f 03 00 fd 5f 03 00 ff-5f 03 00 de 5f 03 00 df
0030:5f 02 01 f7 5f 01 00 00-00 00 00 01 27 01 51 06
0040:5a 00 55 73 65 72 32 00-75 00 73 00 65 00 72 00
0050:32 00 00 00 75 00 73 00-65 00 72 00 32 00 00 00
0060:0c 00 00 06 00 00 00 00-00 00 00 75 00 73 00 65
0070:00 72 00 32 00 00 00 75-00 73 00 65 00 72 00 32
0080:00 40 00 73 00 7a 00 66-00 6b 00 75 00 6b 00 2d
0090:00 64 00 6f 00 6d 00 2e-00 65 00 78 00 74 00 65
00a0:00 73 00 74 00 2e 00 6d-00 69 00 63 00 72 00 6f
00b0:00 73 00 6f 00 66 00 74-00 2e 00 63 00 6f 00 6d
00c0:00 00 00 00 00 00 00 00-00 40 75 00 73 00 65
00d0:00 72 00 32 00 00 00 01-00 00 00 00 00 00 00 00
00e0:00 00 00 00 00 00 00 7c-00 00 00 00 00 dc a7 40
00f0:c8 c0 42 10 1a b4 b9 08-00 2b 2f e1 82 01 00 00
0100:00 00 00 00 00 2f 6f 3d-46 69 72 73 74 20 4f 72
0110:67 61 6e 69 7a 61 74 69-6f 6e 2f 6f 75 3d 45 78
0120:63 68 61 6e 67 65 20 41-64 6d 69 6e 69 73 74 72
0130:61 74 69 76 65 20 47 72-6f 75 70 20 28 46 59 44
0140:49 42 4f 48 46 32 33 53-50 44 4c 54 29 2f 63 6e
0150:3d 52 65 63 69 70 69 65-6e 74 73 2f 63 6e 3d 75
0160:73 65 72 32 00
```

RopId: 0x0e

LogonId: 0x00

InputHandleIndex: 0x08

ColumnCount: 0x000C (COUNT of following **RecipientColumns**)

PidTagObjectType: 0x0ffe0003

PidTagDisplayType:0x39000003

PidTagAddressBookDisplayNamePrintable: 0x39ff001f

PidTagSmtpAddress: 0x39fe001f

PidTagSendInternetEncoding: 0x3a710003

PidTagDisplayTypeEx: 0x39050003

PidTagRecipientDisplayName: 0x5ff6001f

PidTagRecipientFlags: 0x5ffd0003

PidTagRecipientTrackStatus: 0x5fff0003

PidTagRecipientResourceState: 0x5fde0003

PidTagRecipientOrder: 0x5fdf0003

PidTagRecipientEntryId: 0x5ff70102

RowCount: 0x0001 (COUNT of following **ModifyRecipientRows**)

RowId: 0x00000000

RecipientType: 0x01(primary recipient)

RecipientRowSize: 0x0127(bytes in following RecipientRow)

RecipientFlags:0101000100000110 (S,D,Type=X500DN,I,U)

AddressPrefixUsed: 0x5A (present because Type=X500DN)

DisplayType: 0x00 (present because Type=X500DN)

EmailAddress: User2 (present because Type=X500DN)

DisplayName: user2 (present because D is set)

SimpleDisplayName: user2 (present because I is set)

RecipientColumnCount: 0x000C (matches ColumnCount)

StandardPropertyRow:

Flag: 0x00

ValueArray: (property order defined by **RecipientColumns**)

PidTagObjectType:0x00000006

PidTagDisplayType:0x00000000

PidTag7BitDisplayName:user2

PidTagSmtpAddress:user2@szfkuk-dom.extest.microsoft.com

PidTagSendInternetEncoding:0

PidTagDisplayTypeEx:0x40000000

PidTagRecipientDisplayName:user2

PidTagRecipientFlags:0x00000001

PidTagRecipientTrackStatus:0x00000000

PidTagRecipientResourceState:0x00000000

PidTagRecipientOrder:0x00000000

PidTagRecipientEntryId:0x007c and the subsequent 124 (0x7c) bytes

4.7.2 RopModifyRecipients Response Buffer

```
0000: 0e 08 00 00 00 00
```

RopId: 0x0e

InputHandleIndex: 0x08

ReturnValue: 0x000000

4.8 Save Message

After all necessary properties set for the message it was saved. The client sends RopSaveChangesMessage request.

4.8.1 RopSaveChangesMessage Request Buffer

```
0000: 0c 00 00 01 0a
```

RopId: 0x0c

LogonId: 0x00

ResponseHandleIndex: 0x00

InputHandleIndex: 0x01

SaveFlags: 0x0A (KeepOpenReadWrite)

4.8.2 RopSaveChangesMessage Response Buffer

```
0000: 0c 00 00 00 00 00 01 01-00 00 00 00 f0 86 39
```

RopId: 0x0c

ResponseHandleIndex: 0x00

ReturnValue: 0x00000000

InputHandleIndex: 0x01

MessageId: 01 00 00 00 00 f0 86 39

4.9 Releasing Message Object

Finally, the client releases the Message object by using RopRelease, as specified in [MS-OXCROPS].

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the [MS-OXCMSG] protocol. General security considerations pertaining to the underlying **RPC**-based transport apply (see [MS-OXCROPS]).

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following product versions. References to product versions include released service packs.

- Microsoft® Office Outlook® 2003
- Microsoft® Exchange Server 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Exchange Server 2007
- Microsoft® Outlook® 2010
- Microsoft® Exchange Server 2010
- Microsoft® Exchange Server 2010 SP1 Beta

Exceptions, if any, are noted below. If a service pack number appears with the product version, behavior changed in that service pack. The new behavior also applies to subsequent service packs of the product unless otherwise specified.

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

<1> Section 2.2.1.22.4: Outlook 2010 sets PidTagStartDateEtc on a message if it was not defined by the server, using the value of PidTagMessageDeliveryTime if it exists and the value of PidTagCreationTime otherwise.

<2> Section 2.2.2.20: Outlook 2007 correctly detects MacBinary I, MacBinary II, and MacBinary II formats. Outlook 2003 only correctly detects MacBinary I and MacBinary II.

7 Change Tracking

This section identifies changes made to [MS-OXCMSG] protocol documentation between February 2010 and May 2010 releases. Changes are classed as major, minor, or editorial.

Major changes affect protocol interoperability or implementation. Examples of major changes are:

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

Minor changes do not affect protocol interoperability or implementation. Examples are updates to fix technical accuracy or ambiguity at the sentence, paragraph, or table level.

Editorial changes apply to grammatical, formatting, and style issues.

No changes means that the document is identical to its last release.

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

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

- Content removed for template compliance.
- Obsolete document removed.

Editorial changes always have the revision type "Editorially updated."

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

Protocol syntax refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.

Protocol revision refers to changes made to a protocol that affect the bits that are sent over the wire.

Changes are listed in the following table. If you need further information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Revision Type
1.1 Glossary	49614 Added local glossary definitions for "undefined body" and "clear signed body".	N	New content added.
1.3 Overview	Updated the section title.	N	Content updated for template compliance.
2.2.1.11 PidTagImportance	51638 Changed "end-user" back to "end user".	N	Content update.
2.2.1.22 Retention and Archive Properties	54224 Clarified that retention and archive properties apply only to e-mail Message objects.	Y	Content update.
2.2.1.22.4 PidTagStartDateEtc	54224 Added a product behavior note about how Outlook 2010 handles PidTagStartDateEtc.	Y	New content added.
2.2.1.22.5 PidTagRetentionDate	54224 Updated calculation of PidTagRetentionDate.	Y	Content update.
2.2.2.4 PidTagDisplayName	51638 Changed "end-user" back to "end user".	N	Content update.
2.2.2.24 PidTagAttachmentHidden	51638 Changed "end-user" back to "end user".	N	Content update.
4.7.1 RopModifyRecipients Request Buffer	50479 Changed sentence for SimpleDisplayName from "present because S is set" to "present because I is set."	N	Content update.

8 Index