[MS-OXCMAIL]: RFC2822 and MIME to E-Mail Object Conversion Protocol Specification

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability.
Microsoft Corporation	April 25, 2008	0.2	Revised and updated property names and other technical content.
Microsoft Corporation	June 27, 2008	1.0	Initial Release.
Microsoft Corporation	August 6, 2008	1.01	Revised and edited technical content.
Microsoft Corporation	September 3, 2008	1.02	Revised and edited technical content.

Microsoft Corporation	December 3, 2008	1.03	Revised and edited technical content.
Microsoft Corporation	February 4, 2009	1.04	Revised and edited technical content.
Microsoft Corporation	March 4, 2009	1.05	Revised and edited technical content.

Table of Contents

1 Introduction

The RFC2822 and MIME to E-Mail Object Conversion protocol specifies what clients and servers do when they have data in one of these formats, but need it in the other. The process of converting **Message object** data to MIME format is referred to as **MIME generation**, while the reverse process is referred to as **MIME analysis**.

1.1 Glossary

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

address book
ASCII
Attachment object
best body
big-endian
binary large object (BLOB)
Bcc recipient
body part
Cc recipient
code page
character set
charset
Coordinated Universal Time (UTC)
distinguished name (DN)
EntryID
header field
HTML
Internet Message Access Protocol – Version 4 (IMAP4)
.jpg
Mail User Agent (MUA)
message body
message class
Message object
MIME
MIME entity
Out of Office (OOF)
Personal Information Manager (PIM)
plain text
plain text message body
Post Office Protocol - Version 3 (POP3)
property
recipient

remote procedure call (RPC)
RTF
Simple Mail Transfer Protocol (SMTP)
spam
To recipient
Transport Neutral Encapsulation Format (TNEF)
Unicode
universal unique identifier (UUID)
Uniform Resource Identifier (URI)

The following terms are specific to this document:

addressee property group: A group of four related **properties** – display name, **EntryID**, e-mail address type, and e-mail address – that together specify one addressee on a **Message object**.

header: A series of fields (name-value pairs) that supply structured data in an Internet e-mail message, as specified in [RFC2822], or a **MIME entity**. See also: **header field**, **MIME entity**.

Internet Mail Connector Encapsulated Address (IMCEA): A means of encapsulating an e-mail address that is not compliant with [RFC2821] within an e-mail address that is compliant with [RFC2821].

MIME analysis: The process of converting data from an Internet wire protocol to a format suitable for storage by a server or a client.

MIME body: The content of a **MIME entity**, which follows the **header** of the **MIME entity** to which they both belong.

MIME generation: The process of converting data held by a server or a client to a format that is suitable for Internet-standard wire protocols.

MIME reader: An agent that performs **MIME analysis**; it might be either a client or server.

MIME writer: An agent that performs **MIME generation**; it might be either client or server.

primary SMTP proxy address: The **SMTP** email address to be used to designate a message server user in all **SMTP** traffic. Proxy addresses are stored in the user's **address book** entry, in the multi-valued string property **PidTagAddressBookProxyAddresses**. The primary **SMTP** proxy address can be identified by its address type field, which is set to "SMTP" (all upper case). Non-primary **SMTP** proxy addresses have the address type field set to "smtp" (all lower case).

pure MIME message: A **MIME** representation of an e-mail message with no **Transport Neutral Encapsulation Format (TNEF) body part**.

TNEF message: A **MIME** representation of an e-mail message in which attachments and some message properties are carried in a **Transport Neutral Encapsulation Format (TNEF) body part**.

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

[MS-DTYP] Microsoft Corporation, "Windows Data Types", March 2007, http://go.microsoft.com/fwlink/?LinkId=111558.

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

[MS-OXBBODY] Microsoft Corporation, "Best Body Retrieval Protocol Specification", June 2008.

[MS-OXCDATA] Microsoft Corporation, "Data Structures Protocol Specification", June 2008.

[MS-OXCICAL] Microsoft Corporation, " iCalendar to Appointment Object Conversion Protocol Specification", June 2008.

[MS-OXCMSG] Microsoft Corporation, "Message and Attachment Object Protocol Specification", June 2008.

[MS-OXCSPAM] Microsoft Corporation, "Spam Confidence Level, Allow and Block Lists Protocol Specification", June 2008.

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

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

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

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

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

[MS-OXPROTO] Microsoft Corporation, "Exchange Server Protocols Overview", June 2008.

[MS-OXRTFEX] Microsoft Corporation, "Rich Text Format (RTF) Extensions Specification", June 2008.

[MS-OXTNEF] Microsoft Corporation, "Transport Neutral Encapsulation Format (TNEF) Protocol Specification", June 2008.

[MS-RTF] Microsoft Corporation, "Word 2007: Rich Text Format (RTF) Specification, Version 1.9", February 2007, http://go.microsoft.com/fwlink/?LinkId=112393.

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

[RFC1740] Faltstrom, P., Crocker, D., and Fair, E., "MIME Encapsulation of Macintosh files – MACMIME", RFC 1740, December 1994, http://www.ietf.org/rfc/rfc1740.txt.

[RFC1741] Faltstrom, P., Crocker, D., and Fair, E., "MIME Content Type for BinHex Encoded Files", RFC 1741, December 1994, http://www.ietf.org/rfc/rfc1741.txt.

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

[RFC2046] Freed, N. and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996, http://www.ietf.org/rfc/rfc2046.txt.

[RFC2047] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996, http://www.ietf.org/rfc/rfc2047.txt.

[RFC2076] Palme, J., "Common Internet Message Headers", RFC 2076, February 1997, http://www.ietf.org/rfc/rfc2076.txt.

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

[RFC2557] Palme, J., Hopmann, A., and Shelness, N., "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", RFC 2557, March 1999, http://www.ietf.org/rfc/rfc2557.txt.

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

[RFC3282] Alvestrand, H., "Content Language Headers", RFC 3282, May 2002, http://www.ietf.org/rfc/rfc3282.txt.

[RFC3464] Moore, K. and Vaudreuil, G., "An Extensible Message Format for Delivery Status Notifications", RFC 3464, January 2003, http://www.ietf.org/rfc/rfc3464.txt.

[RFC3798] Hansen, T. and Vaudreil, G., "Message Disposition Notification", RFC 3798, May 2004, http://www.ietf.org/rfc/rfc3798.txt.

[RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004, http://www.ietf.org/rfc/rfc3851.txt.

[RFC4646] Phillips, A. and Davis, M., "Tags for the Identification of Languages", RFC 4646, September 2006, http://www.ietf.org/rfc/rfc4646.txt.

1.2.2 Informative References

[MacBin] Apple Corporation, "Macintosh Binary Transfer Format ("MacBinary III") Standard Proposal", http://www.lazerware.com/macbinary/macbinary_iii.html.

[RFC1521] Borenstein, N. and Freed, N., "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, September 1993, http://www.ietf.org/rfc/rfc1521.txt.

[X25] International Telecommunication Union, "ITU-T Recommendation X.25", October 1996, http://www.itu.int/rec/T-REC-X.25-199610-I/en.

1.3 Structure Overview

The representation of electronic mail, calendar items, and other **Personal Information Manager (PIM)** objects by **Message objects** and their properties is described in [MS-OXPROTO] and detailed in [MS-OXOMSG], [MS-OXOCAL], and related specifications.

In contrast, electronic mail, calendar items, and other PIM objects are represented as textual streams when sent over Internet protocols. The textual representation of these streams is commonly referred to as 2822 and/or **MIME** format, as specified by [RFC2822], and [RFC2045] through [RFC2049].

The RFC2822 and MIME to E-Mail Object Conversion protocol specifies how to convert between **Message objects** and MIME-formatted textual streams. The process of converting Message object data to MIME-formatted textual streams is referred to as **MIME generation**, while the reverse process is referred to as **MIME analysis**. Similarly, the agent that performs MIME generation (which might be either a client or server) is referred to as a **MIME writer**, and the agent that performs MIME analysis is referred to as a **MIME reader**.

1.3.1 Data Models

Message objects model e-mail messages and other **PIM** objects after a business memo: there is a single **message body**, with zero or more attachments and zero or more **recipients**. Each Message object has a **message class property** that indicates its type, and an arbitrary collection of properties. Attached messages allow for the nesting of content.

MIME, in contrast, models e-mail messages as a nested set of MIME entities, each of which has **header fields** and a (possibly empty) body. No entity is distinguished as the **message body**. The Content-Type header field indicates the type of each **body part**; other header fields indicate whether a body part is intended as a message body or an attachment. Recipients are modeled by e-mail addresses in certain header fields on the top-level body part. Multipart **body parts** allow for grouping and nesting of content, including attached messages.

The following table shows, at a high level, how the parts of each data model correspond.

MIME	Message object
E-mail address	Recipient
Header field	Property
Body part	Message body
Body part	Attachment

At the next level of detail, some problems become apparent. Because the data models do not match exactly, for each format, it becomes more difficult to convert lower-level items that originate in the other format.

One of the challenges in mapping the Message object content to **MIME** comes from the need to generate human-readable text. Many Message object properties have data types, including binary large objects (**BLOB)**, that do not lend themselves to representation as text. Two solutions are available for these problems:

1. Generate a **pure MIME message**, in which data that does not lend itself to representation in MIME (it is simply omitted from the MIME representation).
2. Generate a **TNEF** message, in which data that does not lend itself to representation in MIME is placed in a TNEF **body part** with a Content-Type of application/ms-tnef. Recipients, plain body text, and top-level header fields are visible to all MIME clients.

Challenges in mapping MIME content to Message objects include distinguishing **message body** from attachments; analyzing multi-part structures that do not fit the MMessage object data model; and mapping header fields or header field parameters that do not have any corresponding property.

Each Message object has a single **charset** (although nested Message objects can have different charsets). MIME, on the other hand, permits the charset of each **header field** and **message body** to be specified separately.

1.4 Relationship to Protocols and Other Structures

Data on the **MIME** side of the conversion is specified by [RFC2822], [RFC2045] through [RFC2049], and related specifications as listed in section 1.2.1 or as referenced from the specifications themselves. Data on the **Message object** side of the conversion is specified by [MS-OXCMSG], [MS-OXOMSG], [MS-OXOCAL], and related specifications as listed in [MS-OXPROTO].

1.5 Applicability Statement

Conversion between **MIME** and **Message object** format is performed in the context of several different protocols. For example:

- Clients and servers perform **MIME generation** for mail outbound to **SMTP**.
- Clients and servers perform **MIME analysis** for mail inbound from SMTP.
- Servers perform MIME generation for Message objects that are downloaded via **POP3** or **IMAP4**. Clients perform MIME generation for messages that are uploaded via IMAP4.
- Servers perform MIME analysis for Message objects that are uploaded via IMAP4. Clients perform MIME analysis for Message objects that are downloaded via POP3 or IMAP4.

1.6 Versioning and Localization

This document covers localization issues in the following area:

- Localization: Localization-dependent content is specified in Sections 2.1.3 and 2.2.3.

Localization is supported by marking messages with locale and **character set** information.

1.7 Vendor-Extensible Fields

[RFC2045] and related RFCs define extensibility mechanisms for **MIME header fields** and content types.

[MS-OXPROPS] defines extensibility mechanisms for **Message object properties** and **message classes**.

2 Structures

The bulk of this specification is divided into two symmetrical parts. Section 2.1 specifies how clients SHOULD set **Message object** properties to produce the desired **MIME** data. Section 2.2 specifies how clients SHOULD create MIME to produce a desired Message object **property** or structure.

A wide variety of possible structures exist for **MIME messages**. One particular structure carries a **Transport Neutral Encapsulation Format (TNEF)** MIME element, which provides a high level of fidelity to original Message object content. All **TNEF messages** have the same structure, as follows:

- At the top level, a **MIME entity** with a Content-Type of "multipart/mixed" that specifies all address elements, as well as the following two child entities:
    - A MIME entity with a Content-Type of "text/plain", that contains a **plain text** rendering of the **message body**.
    - A MIME entity with a Content-Type of "application/ms-tnef" and that contains all attachment content, the **HTML** or **RTF** message body, and any **Message object** properties for which no mapping to MIME **header fields** is defined, encoded as specified in [MS-OXTNEF].

Because a TNEF message is a MIME structure, MIME messages without a TNEF element are sometimes referred to as "pure MIME" to distinguish them from TNEF messages.

2.1 MIME Generation

This section specifies both conversion to pure **MIME** and conversion to **TNEF** from **Message objects**.

When generating a MIME rendering of a Message object, whether pure MIME or TNEF, **MIME writers** SHOULD retrieve all properties of the Message object by issuing one of the following **ROP** sequences (see [MS-OXCROPS]):

- **RopGetPropertiesList** followed by **RopGetPropertiesSpecific**
- **RopGetPropertiesAll**

Clients can explicitly request conversion to pure MIME or TNEF by doing one of the following; a MIME writer SHOULD honor such a client request for message format:

- A client can request conversion to pure MIME for all recipients by setting the value of the **PidTagSendRichInfo property** to FALSE on the Message object, and request conversion to TNEF for all recipients by setting the same property value to TRUE.
- A client can request conversion to pure MIME for an individual recipient by setting the value of the **PidTagSendRichInfo** property to FALSE on that recipient, and request conversion to TNEF for an individual recipient by setting the same property value to TRUE.
- A client can request conversion to pure MIME for an individual one-off recipient by setting the **NoRichInfo** bit in the one-off **EntryID**, as specified in [MS-OXCDATA], and request conversion to **TNEF** by resetting the same bit.

Similarly, when conversion to pure MIME is requested, clients can explicitly request **plain text** or HTML **message body** generation by one of the following means; again, a MIME writer SHOULD honor such a client request for message format:

- A client can request a specific **MIME body** format for all **recipients** by setting the value of the **PidTagSendInternetEncoding** property on the Message object .
- A client can request a specific MIME body format for an individual recipient by setting the value of the **PidTagSendInternetEncoding** property on the recipient.

The value of the **PidTagSendInternetEncoding** property is specified in the following table.

PidTagSendInternetEncoding property value (hex, ABNF)	Desired format

PidTagSendInternetEncoding property value (hex, ABNF)	Desired format
0x00060000 %x00.00.06.00	Plain text only
0x000E0000, %x00.00.0E.00	HTML only <1>
0x00160000, %x00.00.16.00	Both plain text and HTML

2.1.1 Address Elements

In general, address elements are generated in **MIME** only and not in **TNEF**. However, when a **TNEF message** is generated, all address elements of messages that are attached to the top-level message are generated in TNEF only, as specified in [MS-OXTNEF]. This is because there is no **MIME entity** that corresponds to the attached messages; the messages are wholly contained in the TNEF.

MIME writers MUST generate e-mail addresses for MIME **recipients** in compliance with the address requirements specified in [RFC2822]. For example, in cases where a display name is generated in a MIME address **header field**, servers MUST use the encoding specified by [RFC2047] to encode any display name value that has characters that are not allowed in a MIME header field per [RFC2822]. These addresses are always **SMTP** addresses. When a client supports other types of e-mail addresses through the **PidTagAddressType property**, servers SHOULD use **Internet Mail Connector Encapsulated Address (IMCEA)** encapsulation of the e-mail address to form an SMTP address, as specified in section 2.1.1.8 <2>.

Address elements other than recipients, such as From and Sender, are represented in a **Message object** by an **addressee property group** of four properties: display name, address type, e-mail address, and **EntryID**. In subsequent sections, such properties might be referred to as a group. For example, "The **PidTagSentRepresenting property** group" includes the following properties: **PidTagSentRepresentingName**, **PidTagSentRepresentingAddressType**, **PidTagSentRepresentingEmailAddress**, and **PidTagSentRepresentingEntryId**.

2.1.1.1 Recipients

To create a **recipient** in a **MIME** recipient **header field**, clients MUST create a **Message object** recipient with either a **PidTagEntryId** property or both the **PidTagAddressType** and **PidTagEmailAddress properties,** which suffice to fully represent the recipient's e-mail address type and e-mail address. Clients SHOULD, in addition, set **PidTagSmtpAddress**, particularly to save the **SMTP** address when the value of **PidTagAddressType** is not SMTP.

Clients MUST set the **PidTagRecipientType property** value for each recipient as specified by the following table to indicate whether a recipient is a **To recipient**, a **Cc recipient**, or a **Bcc recipient**.

PidTagRecipientType value	Recipient header field
0x00000001	To
0x00000002	Cc
0x00000003	Bcc

When generating MIME or **TNEF**, **MIME writers** SHOULD ignore recipient types other than To, Cc, and Bcc. MIME writers MUST generate one MIME recipient for a **Message object** recipient that has a value of To, Cc, or Bcc. <3> Each MIME recipient MUST be generated in the header field that corresponds to the **PidTagRecipientType** property value, as specified by the **PidTagRecipientType** value table.

Clients SHOULD set the **PidTagDisplayName** property for recipients, where that information is available. MIME writers SHOULD copy the **PidTagDisplayName** property value, when it exists, when generating the display name of an [RFC2822] address specification. The display name MUST be encoded as specified in [RFC2047] when necessary.

MIME writers SHOULD generate the angle-address portion (angle-addr) of an [RFC2822 section 3.4] address specification from addressee properties, used in the following order of preference: **PidTagEntryId**, **PidTagAddressType**/**PidTagEmailAddress**, **PidTagSmtpAddress**. <4> More specifically, MIME writers SHOULD do the following:

1. If **PidTagEntryId** is present and bytes 4-19 are equal to the MUIDEMSAB **UUID** value "{%xdc.a7.40.c8.c0.42.10.1a.b4.b9.08.00.2b.2f.e1.82}", it is an **address book EntryID**. In this case, MIME writers SHOULD look up the address book entry that corresponds to the **DN** that is contained in the EntryID, and use the primary SMTP proxy address that is found on the address book entry. Otherwise, continue to step 2.

(EntryID format is specified in [MS-OXCDATA], and the procedure for looking up **address book** entries is specified in [MS-OXOABK].)

2. If **PidTagEntryId** is present and bytes 4-19 are equal to the MUIDOOP UUID value "{%x81.2b.1f.a4.be.a3.10.19.9d.6e.00.dd.01.0f.54.02}", it is a one-off EntryID. The e-mail type and address are encoded in the EEntryID, as specified in [MS-OXCDATA]. If the e-mail type is SMTP, use this e-mail address; otherwise, continue to step 6.

3. If **PidTagEntryId** is present and bytes 4-19 are some value other than the values that are shown in items 1 and 2, the MIME writer MUST reject the recipient. If **MIME generation** is being done for SMTP, a failure Delivery Status Notification (DSN) MUST be generated for that recipient. The format of a failure DSN is specified in [RFC3464]. The corresponding Message object structure is referred to as a non-delivery receipt; its format is specified in [MS-OXOMSG]. Otherwise, continue to step 4.

4. If both **PidTagAddressType** and **PidTagEmailAddress** are present and **PidTagAddressType** matches SMTP, , continue at step 6 by using the value of **PidTagEmailAddress**. Otherwise, continue to step 5.

5. If the **PidTagSmtpAddress** property is present, use its value. Otherwise, continue to step 6.

6. If an e-mail address and address type are present, whether obtained from **PidTagAddressType** and **PidTagEmailAddress** or from an EntryID, but the address type does not match SMTP, the MIME writer SHOULD attempt **IMCEA** encapsulation of the e-mail address, as specified in section 2.1.1.8.

7. Finally, if all of the previous conditions fail, the MIME writer MUST reject the recipient. If MIME generation is being done for outbound SMTP, a failure DSN MUST be generated for that recipient. The format of a failure DSN is specified in [RFC3464]. The corresponding Message object structure is referred to as a non-delivery receipt; its format is specified in [MS-OXOMSG].

2.1.1.1.1 To and Cc Recipients

To generate a To or Cc **MIME header field**, clients MUST add a **recipient** to the **Message object** and set the **PidTagRecipientType property** to the value that corresponds to the individual recipient type, as specified by the **PidTagRecipientType** value table in section 2.1.1.1.

MIME writers MUST map recipients to the To or Cc MIME header fields as requested by clients. An exceptional situation occurs when generating MIME for an attached DSN message. A DSN message is one that has the following **PidTagMessageClass** value:

```
; The most common values are "REPORT.IPM.Note.NDR" and
"REPORT.IPM.Note.DR"
ReportMsgClass = "REPORT" 1*("." MsgClassToken) (".NDR" / ".DR")

MsgClassToken = ALPHA *(ALPHA / DIGIT)
```

In that case, MIME writers MUST ignore the recipients of the attached message and MUST instead populate the To header field of the attached message by using the **PidTagReceivedRepresenting** property group of the attached message (if it exists), or by using the **PidTagReceivedBy** property group of the attached message.

When generating **TNEF**, MIME writers MUST NOT generate **attRecipTable** for the top-level message. <5> For attached messages, MIME writers MUST copy all recipients, along with all their properties, into the **attRecipTable** TNEF attribute in the TNEF **body part**, as specified in [MS-OXTNEF]. This applies to attached DSN messages as well.

2.1.1.1.2 Bcc recipients

To generate a Bcc **MIME header field**, clients MUST add a **recipient** to the **Message object** and set the **PidTagRecipientType property** value for that recipient to "0x00000003".

When generating a message for outbound submission to **SMTP**, **MIME writers** MUST NOT copy **Bcc recipients** to the MIME Bcc header field. This also applies to the MIME Bcc header field of attached messages. MIME writers MUST NOT copy Bcc recipients to the **TNEF attRecipTable** for attached messages.

When generating a message for **POP3**, **IMAP4**, or similar protocols, MIME writers SHOULD copy Bcc recipients to the MIME Bcc header field. This also applies to the MIME Bcc header field of the attached messages. MIME writers SHOULD copy Bcc recipients to the TNEF recipient table for attached messages.

2.1.1.2 Reply-to

To generate a Reply-To **MIME header field**, clients MUST set the **PidTagReplyRecipientEntries** and the **PidTagReplyRecipientNames properties** to the desired values, as specified in [MS-OXOMSG].

When generating MIME, **MIME writers** MUST generate a Reply-To MIME header field by using the **PidTagReplyRecipientEntries** and the **PidTagReplyRecipientNames** properties. MIME writers SHOULD ignore the **PidTagReplyRecipientNames** value if the count of names does not match the count of entries in the **PidTagReplyRecipientEntries** property. Assuming the counts do match, each entry in the value of the **PidTagReplyRecipientNames** property maps to one display name, and each **EEntryID** in the value of the **PidTagReplyRecipientEntries** property maps to one address, as follows:

1. If bytes 4-19 are equal to the **MUIDEMSAB UUID** value "{%xdc.a7.40.c8.c0.42.10.1a.b4.b9.08.00.2b.2f.e1.82}", it is an **address book** EEntryID. In this case, the MIME writer SHOULD look up the address book entry that corresponds to the **DN** that is contained in the EntryID, and use its primary **SMTP** proxy address.

2. If bytes 4-19 are equal to the **MUIDOOP** UUID value "{%x81.2b.1f.a4.be.a3.10.19.9d.6e.00.dd.01.0f.54.02}", it is a one-off EEntryID. The e-mail type and address are encoded in the EntryID and SHOULD be extracted. If the e-mail type is SMTP, the e-mail address SHOULD be used as is; otherwise, the address MUST be **IMCEA**-encapsulated, as specified in section 2.1.1.8.

When generating **TNEF**, MIME writers SHOULD also copy the values of the **PidTagReplyRecipientEntries** and the **PidTagReplyRecipientNames** properties to the **attMsgProps** attribute in the TNEF **body part**, as specified in [MS-OXTNEF] <6>.

2.1.1.3 From

To generate a From **MIME header field**, clients MUST set the **PidTagSentRepresenting property** group.

When generating MIME, **MIME writers** MUST generate a From header field by using the values of the **PidTagSentRepresenting** property group. The order of preference in that property group MUST be as specified in section 2.1.1.1.

When generating **TNEF**, MIME writers SHOULD also copy the values of the **PidTagSentRepresenting** property group to **attSentFor** and **attMsgProps** in the TNEF **body part**, as specified in [MS-OXTNEF] <7>.

2.1.1.4 Sender

To generate a Sender **MIME header field**, clients MUST set the value of the **PidTagSender property** group.

MIME writers MUST generate a Sender header field by using the values of the **PidTagSenderName** property group. The order of preference in that property group MUST be as specified in section 2.1.1.1. MIME writers SHOULD NOT generate the Sender header field if the **PidTagSenderName** property group and the **PidTagSentRepresenting** property group represent the same **recipient**.

When generating **TNEF**, MIME writers SHOULD also copy the values of the **PidTagSenderName** property group to **attFrom** and **attMsgProps** in the **TNEF body part**, as specified in [MS-OXTNEF] <8>.

2.1.1.5 Return Receipt

To generate a (non-standard) Return-Receipt-To **header field**, protocol clients MUST set the **PidTagOriginatorDeliveryReportRequested property** to TRUE and also MUST set the **PidTagSentRepresenting** property group to the desired values.

MIME writers MUST check the **PidTagOriginatorDeliveryReportRequested property** value first. If the property is not set or the value is FALSE, MIME writers MUST NOT generate the Return-Receipt-To header field.

If the **PidTagOriginatorDeliveryReportRequested** property is set and its value is TRUE and the **PidTagSentRepresenting** property group is set, MIME writers MUST copy the **PidTagSentRepresenting** property group to the Return-Receipt-To header field.

When generating **TNEF**, MIME writers SHOULD copy the values of the **PidTagOriginatorDeliveryReportRequested** and the **PidTagSentRepresenting** property group to **attMsgProps** in the **TNEF body part**, as specified in [MS-OXTNEF] <9>.

2.1.1.6 Read Receipt

To generate a Disposition-Notification-To **MIME header field**, protocol clients MUST set the **PidTagReadReceiptRequested property** to TRUE and also MUST set either the **PidTagReadReceipt** property group or the **PidTagSentRepresenting** property group to the desired values.

MIME writers MUST check the **PidTagReadReceiptRequested** property value first. If the property is not set or the value is FALSE, MIME writers MUST NOT generate the Disposition-Notification-To header field.

If the **PidTagReadReceiptRequested** property is set and its value is TRUE, MIME writers MUST generate the Disposition-Notification-To header field from the **PidTagReadReceipt** property group, if that property group is set. The order of preference in that property group MUST be as specified in section 2.1.1.1. If the **PidTagReadReceipt** property group is not set, protocol servers SHOULD generate the Disposition-Notification-To header field from the **PidTagSentRepresenting** property group.

MIME writers MUST generate the Disposition-Notification-To MIME header field as specified in [RFC3798].

When generating **TNEF**, MIME writers SHOULD also copy the values of the **PidTagReadReceiptRequested**, the **PidTagReadReceipt**, and the **PidTagSentRepresenting** groups of properties to **attMsgProps** in the TNEF **body part**, as specified in [MS-OXTNEF] <10>.

2.1.1.7 Directory Lookups

Clients SHOULD specify the **primary SMTP proxy address** or the **address book** (EX) proxy address in all address elements of a message. But clients MAY use any proxy address. **MIME writers** SHOULD look up each proxy address in the address book, as specified in [MS-OXOABK]. If a matching address book entry is found, MIME writers SHOULD substitute its primary SMTP proxy address for the address specified by the client.

2.1.1.8 IMCEA Encapsulation

When no **SMTP** proxy address is available for an address element, protocol servers SHOULD encapsulate any other address type to produce the required SMTP address, using the **Internet Mail Connector Encapsulated Address (IMCEA)** encapsulation mechanism. <11> The domain part of the encapsulated SMTP address SHOULD be the **MIME** writer's local domain, or another domain that "knows" how to de-encapsulate and deliver to the encapsulated address.

The IMCEA encapsulation mechanism is defined for the address types listed in the following table.

Address type	Value of **PidTagAddressType** or related property
Address book	"EX"
Facsimile	"FAX"
X.400	"X400"

Encapsulated-address = "IMCEA" address-type "-" encoded-address "@" domain
address-type = *VCHAR
domain = dot-atom-text; see [RFC2822] section 3.2.4 for the definition.

encoded-address = * (Escaped-chars/ Normal-chars)

Escaped-chars = (ESCSLASH / ESCCHARS)

; Encoded form for "/" (%x2F) is "_"
ESCSLASH = %x5F

; All OCTETS not ALPHA, DIGIT, or in "-=/"
; These are a "+" and the two hex digits of the OCTET's value.

ESCCHARS = "+" 2(HEXDIG)

; All other characters
Normal-chars= (ALPHA / DIGIT / HYPHEN / EQUALSIGN)
HYPHEN = %x2D
EQUALSIGN = %x3D

Encapsulated addresses MUST NOT include line breaks, and therefore can require longer line lengths than those recommended by [RFC2822]. <12> <13>

2.1.1.9 PidTagAddressType

The value of **PidTagAddressType** is a string that names the messaging system that the address is destined for. It is used to assign responsibility for an e-mail address to the right transport provider. The string value provided by **PidTagAddressType** MUST contain only uppercase alphabetic characters from "A" through "Z", and the numbers from "0" through "9". The value of **PidTagAddressType** is also used designate the correct format for the e-mail address itself, **PidTagEmailAddress**.

If a client tries to compose a message to a user whose address type is not in the server's list of known address types, the message will produce an NDR unless the client itself, acting as the message transfer agent, is able to deliver the message by using an alternate transport.

The following table lists the address types that are known at this time. The common address types include "EX", "SMTP", "X400", and "X500".

Messaging system	**PidTagAddressType** value
Microsoft Exchange Server	"EX"
Internet	"SMTP"
X.400 Message Handling System	"X400"
X.500 Directory Services	"X500"

2.1.2 Envelope Elements

Many **Message object properties** that map to **MIME header field**s have string values. Unless otherwise specified, the string values are simply copied from the property to the header field. When **MIME writers** generate MIME header field values, the encoding specified in [RFC2047] MUST be used where necessary to encode **Unicode** characters.

Likewise, unless otherwise specified, when a MIME message with a **TNEF body part** is being generated, all Message object properties SHOULD be copied to the **attMsgProps** attribute of the TNEF body part, even if there is also a corresponding MIME header field <14>.

2.1.2.1 Message Class

When generating **TNEF**, **MIME writers** MUST copy the value of the **PidTagMessageClass property** to **attMsgProps** in the TNEF **body part**, as specified in [MS-OXTNEF]. In addition, MIME writers SHOULD map the value of the **PidTagMessageClass** property to the **attMessageClass** attribute, as specified in [MS-OXTNEF].

When generating pure MIME, the value of **PidTagMessageClass** SHOULD NOT be copied to MIME messages. Instead, its value is reflected in the structure of the MIME message, as specified in the following table. The MIME structure is indicated by listing the value of the Content-Type **header field**, indented according to how the MIME entities are nested.

PidTagMessageClass value	MIME structure
""`IPM.Note.SMIME.MultipartSigned`", or begins with "`IPM.InfoPathForm.`" and ends with "`.SMIME.MultipartSigned`"	Multipart/signed, as specified in [RFC3851] and [MS-OXOSMIME].
"`IPM.Note.SMIME`", or begins with "`IPM.InfoPathForm.`" and ends with "`.SMIME`"	Application/pkcs7-mime, as specified in [RFC3851] and [MS-OXOSMIME].
"`REPORT.IPM.Note.DR`" or "`REPORT.IPM.Note.NDR`" (other values MAY be substituted for "`IPM.Note`")	As specified in [RFC3464]: multipart/report text/html message/delivery-status <original message structure>

"REPORT.IPM.Note.IPNRN" or "REPORT.IPM.Note.IPNNRN" (other values MAY be substituted for "IPM.Note")	As specified in [RFC3798]: multipart/report text/html message/disposition-notification
Begins with "IPM.Appointment."	Text/calendar, as specified in [RFC2445] and [MS-OXCICAL] <15>.
Begins with "IPM.Schedule.Meeting."	Content mapped to text/calendar, as specified in [RFC2445] and [MS-OXCICAL]. Top-level message structure is multipart/alternative or multipart/mixed, depending on the presence and type of message body and attachments. For details, see section 2.1.3.
"IPM.Note" or any other value	Text/plain, text/html, multipart/alternative, multipart/related, or multipart/mixed, depending on the presence and type of **message body** and attachments. For details, see,section 2.1.3.

2.1.2.2 Content Class

MIME writers SHOULD generate the following values for a Content-Class **header field**, based on the value of the **PidTagMessageClass property <16>**:

PidTagMessageClass value	Content-Class header field value
"IPM.Note.Microsoft.Fax"	"fax"
"IPM.Note.Microsoft.Fax.CA"	"fax-ca"
"IPM.Note.Microsoft.Missed.Voice"	"missedcall"
"IPM.Note.Microsoft.Conversation.Voice"	"voice-uc"

"IPM.Note.Microsoft.Voicemail.UM.CA"	"voice-ca"
"IPM.Note.Microsoft.Voicemail.UM"	"voice"

PidTagMessageClass value begins with	Content-Class header field value
"IPM.Note.Custom."	"urn:content-class:custom.", followed by the value of the **PidTagMessageClass** property with the "IPM.Note.Custom." prefix removed.
"IPM.InfoPathForm."	If the **PidLidInfoPathFormName** property has some value, the Content-Class header field SHOULD be generated with the value of "InfoPathForm.", followed by a string, which is generated as follows: 1) **MIME writer** SHOULD take the value of the **PidTagMessageClass** property, and remove the "IPM.InfoPathForm." prefix. 2) If the remaining string contains a '.' symbol, the value SHOULD be truncated before the period '.'. 3) The value of the **PidLidInfoPathFormName** property SHOULD be appended to this string, preceded by a '.' character.

If the MIME writer was unable to generate a value for the Content-Class MIME header based on the value of the **PidTagMessageClass** property, it SHOULD look up the value of the **PidNameContentClass** property. If this property has a value, that value SHOULD be used as the value of the Content-Class header field; otherwise, a header SHOULD NOT be generated.

2.1.2.3 Unified Messaging Properties

To generate an X-CallingTelephoneNumber **header field**, clients SHOULD set the value of the **PidTagSenderTelephoneNumber property** to the desired value. They MAY instead use **PidNameXSenderTelephoneNumber**. **MIME writers** SHOULD copy either property to the X-CallingTelephoneNumber header field, preferring **PidTagSenderTelephoneNumber**. <17><18>

To generate an X-VoiceMessageDuration header field, clients SHOULD set the value of the **PidTagVoiceMessageDuration** property to the desired value. They MAY instead use **PidNameXVoiceMessageDuration**. MIME writers SHOULD map either property to the X-VoiceMessageDuration header field, preferring **PidTagVoiceMessageDuration**. <19> <20> The value of the **PidTagVoiceMessageDuration** property is a positive valued **PtypInteger32** and MUST be formatted as a decimal string in the header field without sign or separator characters.

To generate an X-VoiceMessageSenderName header field, clients SHOULD set the value of the **PidTagVoiceMessageSenderName** property to the desired value. They MAY instead use **PidNameXVoiceMessageSenderName**. MIME writers SHOULD copy either property to the X-VoiceMessageSenderName header field, preferring **PidTagVoiceMessageSenderName**. <21><22>

To generate an X-FaxNumberOfPages header field, clients SHOULD set the value of the **PidTagFaxNumberOfPages** property to the desired value. They MAY instead use **PidNameXFaxNumberOfPages**. MIME writers SHOULD map either property to the X-FaxNumberOfPages header field, preferring **PidTagFaxNumberOfPages**. <23> <24> The value of the **PidTagFaxNumberOfPages** property is a positive valued **PtypInteger32** and MUST be formatted as a decimal string in the header field without sign or separator characters.

To generate an X-AttachmentOrder header field, clients SHOULD set the value of the **PidTagVoiceMessageAttachmentOrder property** to the desired value. They MAY instead use **PidNameXVoiceMessageAttachmentOrder**. MIME writers SHOULD copy either property to the X-AttachmentOrder header field, preferring **PidTagVoiceMessageAttachmentOrder**. <25><26>

To generate an X-CallID header field, clients SHOULD set the value of the **PidTagCallId** property to the desired value. They MAY instead use **PidNameXCallId**. MIME writers SHOULD copy either property to the X-CallID header field, preferring **PidTagCallId**. <27><28>

2.1.2.4 Arbitrary MIME Header Fields

To generate an arbitrary **header field** on a **MIME** message, a client MUST create a named **property** in the PS_INTERNET_HEADERS property set, with the property name equal to

the header field name and the data type equal to string. The value of this property MUST be set to the desired MIME header field value.

MIME writers MUST use the name and value of such a property to create a header field on the generated MIME message with the corresponding name and value. If necessary, MIME writers MUST encode the header field value as specified in [RFC2047]. But MIME writers MUST NOT create such a header field if a different **Message object** property is already mapped to the same header field, or if the header name begins with one of the reserved name prefixes "X-Microsoft-Exchange-Organization" or "X-Microsoft-Exchange-Forest".

2.1.2.5 Importance

To generate an Importance **header field**, a client MUST set the value of the **PidTagImportance property** as specified in the following table.

PidTagImportance value	Importance header field value
0x00000000	Low
0x00000001	Normal
0x00000002	High

MIME writers MUST map the value of the **PidTagImportance** property to the Importance header field, as specified in the table. MIME writers MAY generate no Importance header field for a **PidTagImportance** value of 1 (normal) or for values other than 0, 1, or 2.

When generating **TNEF**, MIME writers MUST also copy the value of the **PidTagImportance** property to the **attPriority** and **attMsgProps** attributes, as specified in [MS-OXTNEF].

2.1.2.6 Sensitivity

To generate a Sensitivity **header field**, a client MUST set the value of the **PidTagSensitivity property** as specified in the following table.

PidTagSensitivity value	Sensitivity header field value
0x00000000	Normal

PidTagSensitivity value	Sensitivity header field value
0x00000001	Personal
0x00000002	Private
0x00000003	Company Confidential

MIME writers MUST map the value of the **PidTagSensitivity** property to the Sensitivity header field, as specified in the table. MIME writers MAY generate no Sensitivity header field value for a **PidTagSensitivity** value of 0 (normal) or for values other than 0, 1, 2, or 3.

When generating **TNEF**, **MIME writers** MUST also copy the value of the **PidTagSensitivity** property to the **attMsgProps** attribute, as specified in [MS-OXTNEF].

2.1.2.7 Sent Time

To generate a Date **header field**, clients MUST set the value of **PidTagClientSubmitTime** to the desired value. The **property** value MUST be expressed in **UTC**.

MIME writers MUST copy the value of the **PidTagClientSubmitTime** property to the Date header field, formatting it as specified by [RFC2822]. MIME writers SHOULD include hours, minutes, and seconds in the generated Date header field value. MIME writers MAY convert the date and time value from UTC to another time zone of their choice.

If no value is specified for **PidTagClientSubmitTime** when a message is submitted to **SMTP**, MIME writers SHOULD generate a Date header field with a value of the current time.

When generating **TNEF**, MIME writers SHOULD also copy the value of the **PidTagClientSubmitTime** property to the **attDateSent** and **attMsgProps** attributes, as specified in [MS-OXTNEF]. <29>

2.1.2.8 Subject

To generate a Subject **header field**, clients SHOULD set the **PidTagSubjectPrefix** and **PidTagNormalizedSubject** properties on the **Message object**. Clients MAY set the **PidTagSubject property** instead, but in that case, the separation of subject from subject prefix is vulnerable to limitations of the server's parsing procedure, which is specified in section 2.2.2.6.1. Subject property values SHOULD NOT contain line breaks.

MIME writers SHOULD generate the Subject header field by combining the values of the **PidTagSubjectPrefix** and **PidTagNormalizedSubject** properties. <30>

If those two properties are not available, MIME writers MUST copy the value of the **PidTagSubject** property to the Subject header field. MIME writers MAY truncate the subject value; a typical size limit is the first 255 characters. The property value SHOULD NOT be truncated in the middle of a multibyte character.

When generating **TNEF**, MIME writers SHOULD also copy the message subject (however it is obtained) to the **attSubject** and **attMsgProps** attributes, as specified in [MS-OXTNEF]. <31> MIME writers SHOULD also copy the **PidTagSubjectPrefix** and **PidTagNormalizedSubject** properties, with their values, to the **attMsgProps** attribute.

2.1.2.9 Conversation Topic

To generate a Thread-Topic **header field**, clients MUST set the value of the **PidTagConversationTopic property** to the desired value. Clients SHOULD set this property to the same value as **PidTagNormalizedSubject**, with any subject prefix removed, as specified in section 2.2.2.6.1.

MIME writers MUST copy the value of the **PidTagConversationTopic** property to the Thread-Topic header field.

2.1.2.10 Conversation Index

To generate a Thread-Index **header field**, clients MUST set the value of the **PidTagConversationIndex property** to the desired value, as specified in [MS-OXOMSG].

MIME writers MUST copy the value of the **PidTagConversationIndex** property to the Thread-Index header field. The property type is binary; the value MUST be encoded using base64 encoding, as specified in [RFC2045].

2.1.2.11 Message ID

MIME writers SHOULD copy the value of the **PidTagInternetMessageId property** to the Message-ID **header field**. <32> If no value is specified for **PidTagInternetMessageId** when a message is submitted to **SMTP**, MIME writers SHOULD generate a value as specified in [RFC2822].

Clients SHOULD NOT set the **PidTagInternetMessageId** property when submitting a message via **RPC**. As specified in [RFC2822], the value of Message-ID MUST be unique, and for this reason it is normally assigned by servers. Servers MAY overwrite **PidTagInternetMessageId** from a client before submitting the message to SMTP.

Once set, the value of the Message-ID header field and the corresponding property value, **PidTagInternetMessageId**, SHOULD remain constant. MIME writers SHOULD NOT overwrite the value of **PidTagInternetMessageId** when generating **MIME** for protocols such as POP/IMAP.

2.1.2.12 References

To generate a References **header field**, clients MUST set the value of the **PidTagInternetReferences property** to the desired value.

MIME writers MUST copy the value of the **PidTagInternetReferences** property to the References header field.

2.1.2.13 Categories

To generate a Keywords **header field**, clients MUST set the value of the **PidLidCategories property** to the desired values. The type of **PidLidCategories** is multiple strings; each category SHOULD be mapped to a single keyword.

MIME writers SHOULD copy each sub-value of the **PidLidCategories** property to a separate keyword in the Keywords header field, with a comma (U+002C) and space (U+0020) separating each keyword. MIME writers MAY drop the **PidLidCategories** instead of copying it to the Keywords header field, to avoid conflict among different sets of Categories in different organizations.

2.1.2.14 In-Reply-To Message ID

To generate an In-Reply-To **header field**, clients MUST set the value of the **PidTagInReplyToId property** to the desired value.

MIME writers MUST copy the value of the **PidTagInReplyToId** property to the In-Reply-To header field.<33>

2.1.2.15 List Server Properties

To generate a List-Help **header field**, clients MUST set the value of the **PidTagListHelp property** to the desired value.

MIME writers MUST copy the value of the **PidTagListHelp** property to the List-Help header field.

To generate a List-Subscribe header field, clients MUST set the value of the **PidTagListSubscribe** property to the desired value.

MIME writers MUST copy the value of the **PidTagListSubscribe** property to the List-Subscribe header field.

To generate a List-Unsubscribe header field, clients MUST set the value of the **PidTagListUnsubscribe** property to the desired value.

MIME writers MUST copy the value of the **PidTagListUnsubscribe** property to the List-Unsubscribe header field.

2.1.2.16 Language Properties

To generate an [RFC3282] Accept-Language **header field**, clients MUST set the value of the **PidNameAcceptLanguage property** to the desired value.

MIME writers SHOULD copy the value of the **PidNameAcceptLanguage** property to the Accept-Language header field. <34><35> If the **PidNameAcceptLanguage** property is missing, MIME writers SHOULD identify the acceptable locales of the sender's mailbox and write the corresponding language tag, as specified by [RFC4646], as the value of the Accept-Language header field.

To generate an [RFC3282] Content-Language header field, clients MUST set the value of the **PidTagMessageLocaleId** property to the desired locale ID. <36>

MIME writers MUST use the value of the **PidTagMessageLocaleId** property to write the Content-Language header. The value of **PidTagMessageLocaleId** is a Windows LCID (a 32-bit integer value), but the header field value is a language tag, as specified by [RFC4646]. Mapping between LCID and language tag MUST be done as specified in [MS-LCID].

2.1.2.17 Classification Properties

To generate header fields related to message classification, clients MUST set the value of the **PidLidClassified property** to TRUE and the following properties to their desired values: **PidLidClassification**, **PidLidClassificationDescription**, **PidLidClassificationGuid**, **PidLidClassificationKeep**.

When the value of the **PidLidClassified** property is TRUE, **MIME writers** SHOULD copy all classification property values to their corresponding **header fields**, as specified in the following table. <37><38>

Classification property	Classification header field	Property value mapping
PidLidClassified	X-Microsoft-Classified	True => "true" False => no header
PidLidClassificationKeep	X-Microsoft-ClassKeep	Copy string value
PidLidClassification	X-Microsoft-Classification	Copy string value

Classification property	Classification header field	Property value mapping
PidLidClassificationDescription	X-Microsoft-ClassDesc	Copy string value
PidLidClassificationGuid	X-Microsoft-ClassID	True => "true" False => no header

2.1.2.18 Payload Properties

To generate an X-Payload-Provider-Guid **header field**, clients MUST set the value of the **PidTagAttachPayloadProviderGuidString property** to the desired value.

MIME writers SHOULD copy the value of the **PidTagAttachPayloadProviderGuidString** property to the X-Payload-Provider-Guid header field. <39><40>

To generate an X-Payload-Class header field, clients MUST set the value of the **PidTagAttachPayloadClass** property to the desired value.

MIME writers SHOULD copy the value of the **PidTagAttachPayloadClass** property to the X-Payload-Class header field. <41><42>

2.1.2.19 Has Attach

To generate an X-MS-HasAttach **header field**, clients MUST add at least one attachment to the attachment table of the **Message object**.

When the Message object's attachment table contains at least one attachment, **MIME writers** SHOULD generate an X-MS-HasAttach header field with a value of "Yes". When the Message object's attachment table is empty, MIME writers MUST NOT generate an X-MS-HasAttach header field. <43>

2.1.2.20 Auto Response Suppress

To generate an X-Auto-Response-Suppress **header field**, clients MUST set the value of the **PidTagAutoResponseSuppress property** to its desired value.

When the **PidTagAutoResponseSuppress property** has a value of 0 (zero) or -1, **MIME writers** SHOULD map its value to the X-Auto-Response-Suppress header field as shown in the following table. <44><45>

PidTagAutoResponseSuppress property value	X-Auto-Response-Suppress header field value

PidTagAutoResponseSuppress property value	X-Auto-Response-Suppress header field value
0	"None"
-1	"All"

When the **PidTagAutoResponseSuppress** property has a value other than 0 (zero) or -1, MIME writers MUST construct the value of the X-Auto-Response-Suppress header field as follows. For each bit of the value of **PidTagAutoResponseSuppress** that is set (left-hand column), append the string in the right-hand column to the header field value. If the header field value was nonempty, append a comma (U+0032) and space (U+0020) before the new value.

PidTagAutoResponseSuppress property value	X-Auto-Response-Suppress header field value	Description
0x00000001	"DR"	Suppress delivery reports from transport.
0x00000002	"NDR"	Suppress non-delivery reports from transport.
0x00000004	"RN"	Suppress read notifications from receiving client.
0x00000008	"NRN"	Suppress non-read notifications from receiving client.
0x00000010	"OOF"	Suppress **Out of Office (OOF)** notifications.
0x000000020	"AutoReply"	Suppress auto-reply messages other than OOF notifications.

For example, if the value of **PidTagAutoResponseSuppress** is 0x000C, the header field MUST be written as:

```
X-Auto-Response-Suppress: RN, NRN
```

2.1.2.21 Is Auto Forwarded

To generate an X-MS-Exchange-Organization-AutoForwarded **header field**, clients MUST set the value of the **PidTagAutoForwarded property** to TRUE.

If the value of the **PidTagAutoForwarded** property is TRUE, **MIME writers** SHOULD generate the following header field: <46><47>

X-MS-Exchange-Organization-AutoForwarded: true

If the property is absent or the property value is false, a header field SHOULD NOT be generated.

2.1.2.22 Sender Id Status

To generate an X-MS-Exchange-Organization-SenderIdResult **header field**, clients MUST set the value of the **PidTagSenderIdStatus property** to its desired value.

MIME writers SHOULD copy the value of the **PidTagSenderIdStatus** property, which is a **PtypInteger32**, to the X-MS-Exchange-Organization-SenderIdResult header field, formatting it as a string, without separator characters. <48><49>

2.1.2.23 Purported Sender Domain

To generate an X-MS-Exchange-Organization-PRD **header field**, clients MUST set the value of the **PidTagPurportedSenderDomain property** to its desired value.

MIME writers SHOULD copy the value of the **PidTagPurportedSenderDomain** property to the X-MS-Exchange-Organization-PRD header field. <50><51>

2.1.2.24 Spam Confidence Level

To generate an X-MS-Exchange-Organization-SCL **header field**, clients MUST set the value of the **PidTagContentFilterSpamConfidenceLevel property** to its desired value in the range -1 to 10.

MIME writers SHOULD copy the value of the **PidTagContentFilterSpamConfidenceLevel** property, which is a **PtypInteger32**, to the X-MS-Exchange-Organization-SCL header field, formatting it as a decimal numeric string without separator characters. <52><53>

2.1.2.25 Flag Request

To generate an X-Message-Flag **header field**, clients MUST set the value of the **PidLidFlagRequest property** to its desired value.

MIME writers MUST copy the value of the **PidLidFlagRequest** property to the X-Message-Flag header field.

2.1.2.26 TNEF Correlation Key

When creating a new **TNEF** message, **MIME writers** MUST choose a unique key relating the TNEF **body part** to its parent message. (MIME writers SHOULD use the value of **PidTagInternetMessageId** for this purpose.) The chosen value MUST be written in two places:

1. As the value of the X-MS-TNEF-Correlator **header field** on the MIME message.
2. As the value of **PidTagTnefCorrelationKey** in the **attMsgProps** attribute of the TNEF body part itself.

This pair of values SHOULD be used by MIME writers to validate that the top-level message and its TNEF body part do, in fact, belong to each other, and are not (for example) the result of a non-TNEF-aware **Mail User Agent (MUA)** forwarding a message with an attached TNEF body part and retaining the attachment.

2.1.2.27 Received Header Fields

MIME writers SHOULD, under certain circumstances, copy all Received **header fields** from **PidTagTransportMessageHeaders** to the generated **MIME** header. MIME writers MUST NOT copy Received header fields to a MIME message that is bound for **SMTP**, but SHOULD copy the Received header fields to a MIME message that is bound for **POP3** or **IMAP4**. With the exception of headers specifically mentioned elsewhere in section 2.1.2, headers that begin with the reserved name prefixes "X-MS-Exchange-Organization" and "X-MS-Exchange-Forest" SHOULD NOT be copied to **PidTagTransportMessageHeaders**.

Clients SHOULD NOT set the value of **PidTagTransportMessageHeaders**. This **property** value SHOULD be set only upon delivery of a message from SMTP, in which case it SHOULD be set to the MIME message header. <54>

2.1.3 Body Text

When generating **pure MIME**, **MIME writers** MUST generate a single **MIME entity** for the **message body**, and it MUST be the first entity generated. (For **Message objects** without attachments, it SHOULD be the only MIME entity generated.) The MIME entity generated for the message body can have several different structures, some of them fairly complex.

For diagrams of message structure with attachments, and for details about how to determine whether an **Attachment object** represents an inline attached file, see section 2.1.4.

2.1.3.1 Client Actions

To create a **plain text message body** in **MIME**, clients SHOULD set the value of the **PidTagBody property**. Additionally, clients SHOULD set the value of the **PidTagInternetCodepage** property to a **code page** that corresponds to the **charset** that the client wants to appear in MIME. Clients SHOULD NOT create inline **Attachment objects** when the **best body** format of the **Message object** is plain text.

To create an **HTML** message body in MIME, clients SHOULD set the value of the **PidTagHtml** property to the desired HTML text. When this property is set, clients MUST set the value of the **PidTagInternetCodepage** to the code page of the HTML text. (Note that **PidTagHtml** is a Binary property, not a String property.) Clients MAY, instead, set the value of the **PidTagRtfCompressed** property to the desired body text in compressed **RTF** format, depending on the **MIME writer** that is used to convert this text to HTML format. Clients MUST NOT create HTML message text in **Unicode** (UTF-16LE), and the value of **PidTagInternetCodepage** MUST NOT be set to "1200". UTF-32 and UTF-16GE are also not acceptable for this purpose; UTF-7 (code page 65000) and UTF-8 (code page 65001) are acceptable.

To create a multipart/related message body in MIME with HTML body text and inline images, clients SHOULD set the value of the **PidTagHtml** property to the desired HTML text. When this property is set, the value of the **PidTagInternetCodepage** property MUST be set to the code page of the HTML text. (Note that **PidTagHtml** is a binary property, not a string property.) Clients MUST supply a value for either the **PidTagAttachContentId** or the **PidTagAttachContentLocation** property on related file attachments such as images; **PidTagAttachContentId** SHOULD be chosen for this purpose. Depending on the choice of attachment property, inline image links in the HTML body MUST use one of the following:

1. The "cid:" **URI** scheme and a unique content identifier that matches the value of the **PidTagAttachContentId** property on the corresponding Attachment object.
2. A copy of the value of the **PidTagAttachContentLocation** property [see MS-OXCMSG] on the corresponding Attachment object.

Instead of setting the value of the **PidTagHtml** property, clients MAY set the value of the **PidTagRtfCompressed** property and include OLE attachments, depending on the protocol server to convert the RTF text to HTML and the static renderings of the OLE attachments to image attachments. For details, see section 2.1.3.7.

For **plain text** messages, clients SHOULD write the value of the **PidTagBody** property in **Unicode** and SHOULD set the value of the **PidTagInternetCodepage** property to the code page that matches the sender's preferred charset. When generating a MIME element for the

plain text body, **MIME writers** MUST map this code page to a charset name, MUST convert the Unicode text into that charset, and MUST write that charset name to the value of the *charset* parameter of the Content-Type **header field**. The plain text MIME element generated for a **TNEF** message SHOULD be treated in the same way.

For HTML messages, clients SHOULD write the value of the **PidTagHtml** property by using text in the sender's preferred charset. Clients MUST set the value of the **PidTagInternetCodepage** property to the code page that corresponds to the preferred charset. Clients MUST NOT use UTF-16 (code page 1200) as the preferred charset. If the HTML document contains a content-type meta tag, its *charset* parameter value SHOULD match the preferred charset.

When generating a MIME element or elements for an HTML message body, MIME writers MUST map the value of the **PidTagInternetCodepage** property to a charset name, MUST write the MIME element body in that charset, and MUST write that charset name as the value of the Content-Type header field's *charset* parameter. If the HTML document contains a content-type meta tag, its *charset* parameter value SHOULD match the Content-Type header field's *charset* parameter value.

For **RTF** messages, clients SHOULD write the value of the **PidTagRtfCompressed** property by using text in the sender's preferred charset. Clients SHOULD set the value of the **PidTagInternetCodepage** property to the code page that corresponds to the preferred charset. The preferred charset MUST NOT be UTF-16 (code page 1200). MIME writers MUST NOT rely on the value of the **PidTagInternetCodepage** property, but treat it as a preference; MIME writers SHOULD instead rely on the value of one or more \ansicpg elements in the RTF stream, as specified in [MS-RTF], to determine the actual body code page.

When generating a MIME element or elements for an RTF message body, MIME writers SHOULD convert the RTF text to plain text or HTML, SHOULD map the body code page to a charset name, SHOULD write the **MIME element** body in that charset, and SHOULD write that charset name as the value of the Content-Type header field's *charset* parameter <55>

Even if a **Message object** has no body, clients SHOULD set the value of the **PidTagInternetCodepage** property to indicate a preferred charset for header field text, to be used in [RFC2047] encoding.

When generating header fields for a **MIME entity**, it might be necessary to encode the characters as specified in [RFC2047]. MIME writers SHOULD use the same charset for all header fields and the message body. Attachments that are themselves messages are independent and can have a different charset.

2.1.3.2 Message Body in TNEF

When generating **TNEF**, **MIME writers** SHOULD identify the "best body" **property** of the **Message object**, as specified in [MS-OXBBODY], and copy its value to the **attMsgProps**

attribute of the TNEF **body part**. MIME writers MUST also place a **plain text** version of the **message body** in the first child body part of the **TNEF message**, as specified in section 2, generating plain text from the value of the "best body" property if necessary. Finally, when the **best body** is plain text, MIME writers SHOULD also write a matching value for the **PidTagRtfCompressed** property (generating it if necessary) to the **attMsgProps** attribute of the TNEF body part.<56>

2.1.3.3 Simple Plain Text Message Body

When the **best body** format type is **plain text**, **MIME writers** SHOULD generate a single **MIME entity** with the value of its Content-Type **header field** set to text/plain.

The *charset* parameter value of this MIME entity's Content-Type header field SHOULD be set to a **charset** that corresponds to the value of the **PidTagInternetCodepage property** (a **code page** number). If there is no **PidTagInternetCodepage** property, the value of the **PidTagMessageCodepage** property MAY be used instead, but in that case it SHOULD first be mapped from a Windows code page to the corresponding Internet code page. MIME writers SHOULD verify that the plain text, which is stored as UTF-16, can actually be encoded in this charset and SHOULD, if necessary, choose a different charset that can in fact encode the entire **message body**; the code page properties express a preference rather than a requirement.

The value of the **PidTagBody** property MUST be written to the content of the text/plain MIME element, after being converted to the chosen charset.

2.1.3.4 HTML Text Message Body Without Inline Attachments

When the **best body** format type is **HTML** and no inline **Attachment objects** exist, **MIME writers** SHOULD generate a **MIME entity** with multipart/alternative for the value of its Content-Type **header field**, and with the following two child entities:

1. The first child entity MUST have "`text/plain`" for the value of its Content-Type header field. Its body SHOULD be **plain text** generated from the value of the **PidTagHtml property**. The body can instead be copied from the value of the **PidTagBody** property, assuming that **PidTagBody** contains substantially similar text to **PidTagHtml**. <57>

2. The second child entity MUST have "`text/html`" for the value of its Content-Type header field. Its body MUST be the value of the **PidTagHtml** property.

The *plain text* and *charset* parameters SHOULD be processed as specified in section 2.1.3.3. HTML text MAY be processed in exactly the same way, or characters that do not fit the preferred **charset** MAY instead be encoded within the HTML.

2.1.3.5 HTML Text Message Body from RTF Without Inline Attachments

When the **best body** format type is **RTF** and no inline (OLE) **Attachment objects** exist, **MIME writers** SHOULD generate a multipart/alternative **MIME entity** with the following two child entities:

1. The first child entity MUST have "`text/plain`" for the value of its Content-Type **header field**. Its body SHOULD be **plain text** generated from the value of the **PidTagRtfCompressed property**, but MAY instead be copied from the value of the **PidTagBody** property, assuming that it contains substantially similar text.

2. The second child entity MUST have "`text/html`" for the value of its Content-Type header field. Its body MUST be **HTML** text. The HTML text SHOULD be generated from the value of the **PidTagRtfCompressed** property, but MAY instead be copied form the value of the **PidTagHtml** property, assuming that it contains substantially similar text.

The *text* and *charset* parameters SHOULD be processed as specified in section 2.1.3.4.

2.1.3.6 HTML Text Message Body with Inline Attachments

When the **best body** format type is **HTML** and inline **Attachment objects** exist, **MIME writers** SHOULD generate a **MIME entity** with "multipart/related" for the value of its Content-Type **header field** and two or more child elements, as follows:

1. The first child entity MUST be a "`multipart/alternative`" structure, exactly as specified in section 2.1.3.4.

2. Subsequent child entities MUST be generated from the **Message object's** inline attachments. A child entity MUST be generated if and only if the Attachment object is marked as specified in section 2.1.4.1.

MIME writers SHOULD verify that the HTML text actually contains a reference to each inline Attachment object, either by its **PidTagAttachContentId** or **PidTagAttachContentLocation property**, as specified in section 2.1.3.1. If the HTML text contains no such reference, the MIME writer SHOULD consider this Attachment object as not inline and generate its MIME entity as a peer of the multipart/related MIME entity, instead of as its child.<58>

2.1.3.7 HTML Text Message Body from RTF with Inline (OLE) Attachments

When the **best body** format type is **RTF** and inline (OLE) **Attachment objects** exist, **MIME writers** SHOULD generate a **MIME entity** with multipart/related for the value of its Content-Type **header field** and three or more child entities, as follows:

1. The first child entity MUST be a multipart/alternative structure, exactly as specified in section 2.1.3.5.

2. Subsequent child entities MUST be generated from the **Message object's** inline attachments. Each entity MUST be generated as specified in section 2.1.4.4, because inline Attachment objects in RTF messages are always OLE attachments.

2.1.3.8 Calendar Items and Meeting Messages

A **Message object** is a calendar item when the value of **PidTagMessageClass** starts with "`IPM.Appointment.`" or equals"`IPM.Appointment`". A Message object is a meeting message when the value of **PidTagMessageClass** starts with "`IPM.Schedule.Meeting.`" or equals "`IPM.Schedule.Meeting`". Clients SHOULD create items of these types with a **best body** format type of **RTF**. Clients MAY use a **plain text** body instead, but SHOULD NOT create calendar items or meeting messages with a best body format type of **HTML**.

Each of the leaf **MIME** entities specified in this section SHOULD use UTF-8 as its **charset**, as specified in [RFC2445].

2.1.3.8.1 Plain Text Calendar Message

When the **best body** format type of a calendar item or meeting message is **plain text**, **MIME writers** SHOULD generate a **MIME entity** with multipart/alternative for the value of its Content-Type **header field** and two child entities, as follows:

1. The first child entity MUST have "`text/plain`" for the value of its Content-Type header field, and its content MUST be copied from **PidTagBody**.

2. The second child entity MUST have text/calendar for the value of its Content-Type header field, and its content MUST be generated as specified in [MS-OXCICAL].

2.1.3.8.2 Calendar Message Without Inline Attachments

When the **best body** format type of a calendar item or meeting message is **RTF** and there are no inline attachments, **MIME writers** SHOULD generate a **MIME entity** with "`multipart/alternative`" for the value of its Content-Type **header field** and three child entities, as follows:

1. The first child entity MUST have "`text/plain`" for the value of its Content-Type header field. Its content SHOULD be **plain text** generated from the value of the **PidTagRtfCompressed property**, but MAY instead be copied from the value of the **PidTagBody** property, assuming that it contains substantially similar text.

2. The second child entity MUST have "`text/html`" for the value of its Content-Type header field. Its content SHOULD be **HTML** text generated from the value of the

PidTagRtfCompressed property, but MAY instead be copied from the value of the **PidTagHtml** property, assuming that it contains substantially similar text. <59>

3. The third child entity MUST have "`text/calendar`" for the value of its Content-Type header field, and its content MUST be generated as specified in [MS-OXCICAL].

2.1.3.8.3 Calendar Message with Inline Attachments

When the **best body** format type of a calendar item or meeting message is **RTF** and there are inline attachments, **MIME writers** SHOULD generate a **MIME entity** with "`multipart/related`" for the value of its Content-Type **header field** and two or more child entities, as follows:

1. The first child entity MUST be a multipart/alternative structure generated as specified in section 2.1.3.8.2.
2. Subsequent child entities MUST be generated from the **Message object's** inline attachments. Each entity MUST be generated as specified in section 2.1.4.1.

2.1.4 Attachments

Each **Attachment object** in a **Message object** represents one attachment. **MIME writers** SHOULD classify Attachment objects based on the value of **PidTagAttachMethod property**, as specified in the following table.

PidTagAttachMethod value	Attachment Object Classification
5	Embedded message attachments
6	OLE attachments
All other values	Ordinary file attachments

Note that ordinary file attachments might contain additional Macintosh-specific data. These attachments require special handling, as specified in section 2.1.4.3. Additionally, MIME writers SHOULD classify Attachment objects as inline or not inline, as specified in section 2.1.4.1.

MIME writers SHOULD generate different MIME structures for the message depending on the presence of inline and non-inline attachments, as specified in the following three examples:

1. If both inline and non-inline attachments are present, MIME writers SHOULD generate the structure shown in Example 1. Figure 1 shows a graphical representation of the actual message structure.

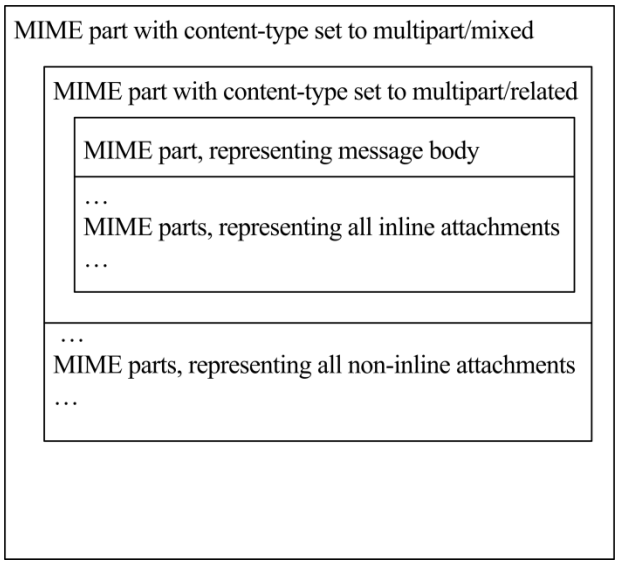

Figure 1: Inline and non-inline attachments present

Example 1:

```
From: <user1@example.com>
To: <user2@example.com>
Subject: Example with inline and non-inline attachments.
Date: Mon, 10 Mar 2008 14:36:46 -0700
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="simple boundary 1"

--simple boundary 1
Content-Type: multipart/related; boundary="simple boundary 2"

--simple boundary 2
Content-Type: multipart/alternative; boundary="simple boundary 3"

--simple boundary 3
Content-Type: text/plain

...Text without inline reference...
--simple boundary 3
Content-Type: text/html
```

```
...Text with inline reference...
--simple boundary 3--
--simple boundary 2
Content-Type: image/png; name="inline.PNG"
Content-Transfer-Encoding: base64
Content-ID: <6583CF49B56F42FEA6A4A118F46F96FB@example.com>
Content-Disposition: inline; filename="Inline.png"

...Attachment data encoded with base64...
--simple boundary 2--

--simple boundary 1
Content-Type: image/png; name=" Attachment "
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Attachment.png"

...Attachment data encoded with base64...
--simple boundary 1--
```

2. If only inline attachments are present, MIME writers SHOULD generate the structure shown in Example 2. Figure 2 shows a graphical representation of the actual message structure.

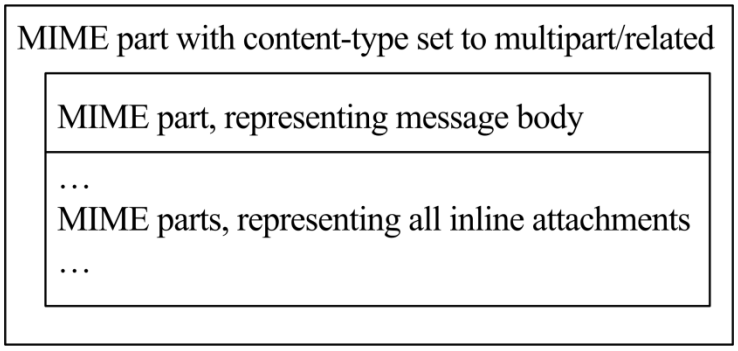

Figure 2: Only inline attachments present

Example 2:

```
From: <user1@example.com>
To: <user2@example.com>
Subject: Example with inline attachment.
Date: Mon, 10 Mar 2008 14:36:46 -0700
MIME-Version: 1.0
Content-Type: multipart/related; boundary="simple boundary"

--simple boundary
```

```
Content-Type: text/html;

...Text with reference...

--simple boundary
Content-Type: image/png; name="inline.PNG"
Content-Transfer-Encoding: base64
Content-ID: <6583CF49B56F42FEA6A4A118F46F96FB@example.com>
Content-Disposition: inline; filename=" inline.png"

...Attachment data encoded with base64...
--simple boundary--
```

3. If only non-inline attachments are present, MIME writers SHOULD generate the structure shown in Example 3. Figure 3 shows a graphical representation of the actual message structure.

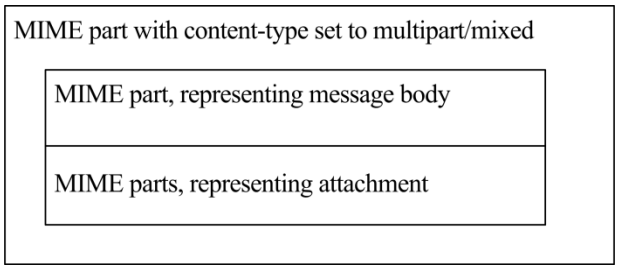

Figure 3: Only non-inline attachments present

Example 3:

```
From: <user1@example.com>
To: <user2@example.com>
Subject: Example with non-inline attachment.
Date: Mon, 10 Mar 2008 14:36:46 -0700
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="simple boundary"

--simple boundary
Content-Type: text/plain;

...Text without reference...

--simple boundary
Content-Type: image/png; name=" Attachment"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Attachment.png"
```

```
...Attachment data encoded with base64...
--simple boundary--
```

2.1.4.1 Inline Attachments

Clients SHOULD NOT create inline attachments if the **best body** text format is **plain text**. **MIME writers** SHOULD ignore **PidTagAttachFlags** and other indications that an attachment is inline for plain text messages. Likewise, clients SHOULD NOT designate attached **Message objects** as inline, and MIME writers SHOULD NOT treat attached Message objects as inline.<60>

2.1.4.1.1 Inline Attachments in RTF Messages

If the **best body** text format is **RTF**, **MIME writers** SHOULD treat all OLE attachments, and only OLE attachments, as inline attachments <61>. OLE attachments have 0x0000006 for the value of **PidTagAttachMethod**.

RTF text does not contain explicit references to inline attachments, as **HTML** text does. Instead, the position of an inline attachment in the RTF text is indicated by an "`\objattph`" tag; clients MUST insert such a tag into the RTF text for each inline attachment, as specified in [MS-OXRTFEX]). Clients MUST also set the value of the **PidTagRenderingPosition property** to indicate the order of inline attachments: the attachment with the lowest value of this property matches the first "`\objattph`" tag; the next lowest matches the second "`\objattph`" tag, and so on. Finally, clients SHOULD set the 0x00000000 bit in the value of the **PidTagAttachFlags** property to indicate that it is inline. MIME writers MUST sort inline attachments by the value of **PidTagRenderingPosition** when converting RTF text with inline attachments to HTML, and map the RTF "`\objattph`" tag to an HTML IMG tag at the corresponding position in the generated HTML.

2.1.4.1.2 Inline Attachments in HTML Messages

To mark an **Attachment object** in a message the **best body** text format for which is **HTML** as inline, clients MUST do the following:

1. Set bit 3 (0x00000004) in the value of the Attachment object's **PidTagAttachFlags property** to TRUE.
2. Set the value of either **PidTagAttachContentId** (preferred) or **PidTagAttachContentLocation** on the Attachment object. If **PidTagAttachContentLocation** is used, **PidTagAttachContentBase** MAY be set to fully qualify a relative **URI** in **PidTagAttachContentLocation**. For details, see [RFC2557].

3. Include a tag that refers to the URI specified in (2) in the HTML message text. If **PidTagAttachContentId** is used, the URI MUST use the "cid:" scheme.

MIME writers SHOULD NOT rely entirely on bit 0x00000004 of the **PidTagAttachFlags** property value to be set correctly for all attachments. Instead, MIME writers SHOULD verify all three conditions specified when deciding whether to treat an attachment as inline. <62>

2.1.4.2 Attached Files

This section describes **MIME generation** for ordinary attachments without Macintosh-specific data.

The remainder of this section concerns generating attachments for **pure MIME messages**. When generating a **TNEF** message, all attachment data MUST be written to the TNEF **body part**, as specified in [MS-OXTNEF].

2.1.4.2.1 File Name

For the file name in a **MIME** representation of an attached file, **MIME writers** SHOULD use the value of the **PidTagAttachLongFilename property**. If this value is not available, MIME writers SHOULD use the value of the **PidTagAttachFilename** property, and MAY use an empty string if this value is also not available. The attached file name SHOULD be written to several different MIME headers, as specified in the next section.

If a file extension is needed for mapping the attachment content type, it SHOULD be obtained by copying all characters after the last "." (U+002E) character in the file name.

2.1.4.2.2 Content-Type, Content-Description, Content-Disposition

MIME writers SHOULD determine the primary value of the Content-Type **header field** for an attached file by using the following steps:

1. Acquire the value of the **PidTagAttachMimeTag property**. If this value is not available, MIME writers SHOULD determine the Content-Type by mapping it from the file extension (which SHOULD be determined from the attachment file name, as specified in section 2.1.4.2.1), or by examining the file content itself. As a last resort, the MIME writer MUST use "`application/octet-stream`".
2. If the value acquired in the previous step does not match requirements for MIME Content-Type, as specified in [RFC2045], or if the value represents any multipart Content-Type, or if the value matches one of the following values, MIME writers SHOULD replace it with "`application/octet-stream`":
    - application/applefile
    - application/mac-binhex40
    - message/rfc822

The value acquired as a result is then used as the value of the Content-Type MIME header field. MIME writers SHOULD also generate the *name* parameter for this header field, by using the attachment file name (determined as specified in section 2.1.4.2.1) as a value.

MIME writers SHOULD generate a Content-Description header field by using the value of the **PidTagDisplayName** property. <63> If the property has no value, an empty header field MAY be generated. The value of the Content-Description header field SHOULD be encoded as specified in [RFC2047] when applicable.

The value for Content-Disposition header field SHOULD be generated based on whether the attachment is inline or not, as specified in section 2.1.4.1. For inline attachments, the value MUST be "`inline`", and for non-inline attachments, the value MUST be "`attachment`". MIME writers SHOULD generate the following parameters for this header field:

- *filename*: the attachment file name determined as specified in section 2.1.4.2.1 MUST be used as a value.
- *size*: **PidTagAttachSize** property value SHOULD be used as a parameter value. The size parameter SHOULD be generated only if this property value is available and greater than 0 (zero). <64><65>
- *creation-date*: **PidTagCreationTime** property value SHOULD be used as the parameter value; if the property value is not available, the current time SHOULD be used. In either case, the creation time SHOULD be converted from **UTC** to a local time zone of the MIME writer's choice and formatted as specified in [RFC2822]. <66><67><68>
- *modification-date*: **PidTagLastModificationTime** property value SHOULD be used as the parameter value; if the property value is not available, the current time SHOULD be used. In either case, the modification time SHOULD be converted from UTC to a local time zone of the MIME writer's choice and formatted as specified in [RFC2822]. <69><70><71>

2.1.4.2.3 Content-ID, Content-Location, Content-Base

MIME writers SHOULD generate a Content-ID MIME **header field** if the value of the **PidTagAttachContentId** contains non-whitespace characters. All trailing and leading whitespace characters SHOULD be removed from this value. If the resulting value does not start with '<' (U+003C), or does not end with '>' (U+003E), it SHOULD be enclosed in angle brackets. The resulting string becomes the value of the Content-ID header field.

MIME writers SHOULD generate a Content-Location MIME header field if the **PidTagAttachContentLocation property** contains a value that is a valid **URI**. This value SHOULD be copied to the value of the Content-Location header field.

MIME writers SHOULD generate a Content-Base MIME header field if the **PidTagAttachContentBase** property contains a value that is a valid absolute URI. This value SHOULD be copied to the value of the Content-Base header field.

2.1.4.2.4 Content-Transfer-Encoding, MIME Part Body

The server SHOULD use base64 (see [RFC4648]) as encoding for all ordinary file attachment **MIME** part bodies. As specified in [RFC2045], this also means that the server SHOULD correspondingly generate the Content-Transfer-Encoding MIME **header field**, and set its value to "`base64`".

MIME writers MUST use the value of the **PidTagAttachDataBinary property** to generate the **MIME entity** body for this attachment. If the property does not exist or has 0 (zero) length, an empty MIME entity body SHOULD be generated.

2.1.4.3 MacBinary Attached Files

For interoperability with Macintosh-based mail clients, sometimes it is useful to encode message attachments in **MIME** by using one of the following Content-Types that are recommended for use in Macintosh environment:

- application/applefile, as specified in [RFC1740].
- application/mac-binhex40, as specified in [RFC1741].
- multipart/appledouble, as specified in [RFC1740].

MIME writers SHOULD generate multipart/appledouble, as this MIME type is recommended by [RFC1740] for use in most cases. <72>

As specified in [RFC1740], the multipart/appledouble MIME part contains two sub-parts: a header part, with a Content-Type of "`application/applefile`", and a data part that contains actual file data (with Content-Type set to the value that corresponds to the actual MIME type of the file that is encoded).

To trigger encoding of an **Attachment object** as multipart/appledouble, clients MUST set **property** values on the Attachment object as follows:

1. The value of the **PidTagAttachMethod** property MUST be "0x00000001" (file attachment).
2. The value of the **PidTagAttachEncoding** property MUST be the following byte string (expressed in hexadecimal): "%x2A.86.48.86.F7.14.03.0B.01".
3. The attachment content, which is the value of the **PidTagAttachDataBinary** property, MUST be encoded in MacBinary format as specified in [MacBin].

MacBinary is a way of serializing all attributes of a Macintosh file, including both data and resource forks, into a single stream. MacBinary format is specified in [MacBin] and the elements relied upon in this specification are summarized very briefly by the following two tables. What follows is intended to specify server behavior with respect to MacBinary data; it is not normative with respect to the MacBinary format itself.

MacBinary data field	Length	Description
MacBinary header	128 bytes.	See more detail later in this section.
Secondary header data	Length is specified in bytes 120:121 of MacBinary header.	SHOULD be ignored by MIME writers. <73>
Data fork	Length is specified in bytes 83:86 of MacBinary header; begins on an even multiple of 128 bytes.	Contents of the file.
Resource fork	Length is specified in bytes 87:90 of MacBinary header; begins on an even multiple of 128 bytes.	Resources associated with the file.
Get Info comment	Length is specified in byte 99 of MacBinary header.	SHOULD be ignored by MIME writers. <74>

Byte offset and length	Value
Byte 0	Old version number, MUST be zero.
Byte 1	Length of file name, MUST be less than 64.
Bytes 2 : 64	File name, in us-ascii **charset**; characters beyond the length specified in byte 1 MUST be ignored.

Byte offset and length	Value
Byte 65 : 68	File type, signed integer.
Byte 69 : 72	File creator, signed integer.
Byte 74	Pad, MUST be 0 (zero).
Byte 82	Pad, MUST be 0 (zero).
Bytes 83 : 86	Data fork length, signed 32-bit integer in **big-endian** format.
Bytes 87 : 90	Resource fork length, signed 32-bit integer in big-endian format.

MIME writers MUST create a **MIME entity** with a Content-Type value of "`multipart/appledouble`", as specified in [RFC1740]. MIME writers SHOULD NOT write a *name* parameter for the Content-Type header in this MIME part. (This parameter is optional, as specified in [RFC1740].) As specified in [RFC1740], all additional information (other than the file contents) for a file that is to be transmitted by using the multipart/appledouble MIME Content-Type SHOULD be put into a sub-part with Content-Type application/applefile.

If the **Attachment object's PidNameAttachmentMacInfo** property has a value, MIME writers MUST use it as the body of the application/applefile **body part**. The value of this property SHOULD be application/applefile data, as specified in [RFC1740] and further detailed in section 2.2.4.2.2, but containing only the header and resource fork sections.

If the Attachment object's **PidNameAttachmentMacInfo** property has no value, MIME writers SHOULD generate the body of the application/applefile body part from the resource fork and header data present in the **MacBinary** structure from the **PidTagAttachDataBinary** property, by using the mappings specified in section 2.2.4.2.2.

As specified in [RFC1740], the actual contents (or data fork) of the file that is to be transmitted by using the multipart/appledouble MIME Content-Type MUST be put into a second MIME sub-part under multipart/appledouble.

This MIME part is written out in the same way as in the case of an ordinary file attachment, with the following exceptions:

1. MIME writers MUST generate this part's **MIME body** by extracting only the file's data fork from the **MacBinary** structure in the **PidTagAttachDataBinary** property on the attachment, instead of just using raw data from this property.

2. MIME writers SHOULD copy the value of the **PidNameAttachmentMacContentType** property to the attachment body part's Content-Type **header field**.

If **PidNameAttachmentMacContentType** has no value, MIME writers SHOULD write Content-Type: "`application/octet-stream`". An application/octet-stream type SHOULD also be written if **PidNameAttachmentMacContentType** has one of the following values:

- message/rfc822
- application/applefile
- application/mac-binhex40
- any multipart content-type

2.1.4.4 OLE Attachments

This section describes the generation of **MIME entities** that correspond to OLE attachments. An **Attachment object** is an OLE attachment if its **PidTagAttachMethod property** is set to 0x00000006.

MIME writers SHOULD generate a MIME part with "image/jpeg" for the value of its Content-Type **MIME header field** to represent an OLE attachment in MIME. MIME writers SHOULD generate a description string for an OLE attachment, by using the value of the **PidTagDisplayName** property, but ensuring that this value ends with "**.jpg**". The description string SHOULD be used as the *name* parameter of the Content-Type MIME header field, and the value of the Content-Description MIME header field SHOULD be generated with the same value.

A Content-Disposition header field SHOULD be generated in the same way as for ordinary file attachments, with the following exceptions:

1. The *size* parameter SHOULD NOT be generated.

2. The *filename* parameter value SHOULD be set to description string (see section 2.1.4.2.1).

The rest of MIME part headers SHOULD be generated in the same way as for ordinary file attachments, as specified in section 2.1.4.1.

OLE attachments SHOULD NOT have the **PidTagAttachDataBinary** property set, so MIME part body cannot be generated in the same way as for ordinary file attachments. Instead, the **PidTagAttachDataObject** property SHOULD be used. This property SHOULD

contain a static rendition of an OLE object in Windows metafile format, as specified in [MS-WMF]. **MIME writers** SHOULD use this data to generate a JPEG image that represents this OLE object, and generate the MIME part body by using this image data. If image generation fails, the server SHOULD use some default image. <75>

2.1.4.5 Embedded Message Attachments

This section describes the generation of **MIME** entities that correspond to embedded message attachments. An attachment MUST be considered by **MIME writers** to be an embedded message attachment if the value of its **PidTagAttachMethod property** is "0x00000005". MIME writers SHOULD generate a **MIME entity** with the Content-Type **header field** set to "`message/rfc822`" (without parameters being generated). No other MIME headers SHOULD be generated. Instead, MIME writers SHOULD use properties of the embedded message to generate a pure MIME representation of this message, exactly as specified for ordinary messages, and use this data as the content of the message/rfc822 MIME entity. This MIME representation SHOULD be generated exactly as specified for ordinary messages, with the following exception: when writing MIME message headers by using PS_INTERNET_HEADERS properties, as specified in section 2.1.2.4, properties whose names begin with "X-MS-Exchange-Organization-" or "X-MS-Exchange-Forest-" SHOULD NOT be excluded from **MIME generation** (as they are for ordinary messages).

2.2 MIME Analysis

This section specifies both conversion from pure **MIME** to **Message objects**, and from **TNEF** to Message objects. The agent that performs the conversion is referred to as a **MIME reader** for clarity, because both clients and servers perform this conversion for different protocols.

As a general rule, when data occurs both in MIME and in a TNEF **body part**, the version found in MIME is to be preferred. The **message body** is an exception to this rule: the **plain text** rendering found in MIME SHOULD NOT be used in preference to a richer (**HTML** or **RTF**) rendering found in TNEF. As an implementation guideline, MIME readers MAY process the TNEF body part before processing the remaining MIME data so that data from MIME overwrites the conflicting data from TNEF.

2.2.1 Address Elements

Most **MIME** address elements correspond to a group of four properties in the **Message object**. The MIME address element itself has three parts, as specified in [RFC2822]: display name, comment, and e-mail address. The four **properties** are **DisplayName**, **EmailAddress**, **AddressType**, and **EntryId**. For a recipient in a Message object, the four properties are referred to as the Recipient property group, the members of which are the following:

- **PidTagDisplayName**

- **PidTagEmailAddress**
- **PidTagAddressType**
- **PidTagEntryId**

For other address elements in a Message object, the four properties are grouped by name. For example, the four properties that correspond to the From **header field** are the following:

- **PidTagSentRepresentingName**
- **PidTagSentRepresentingEmailAddress**
- **PidTagSentRepresentingAddressType**
- **PidTagSentRepresentingEntryId**

Collectively, these four properties are referred to as the **PidTagSentRepresenting property** group.

2.2.1.1 Mapping Internet Address Elements to a Property Group

In general, **MIME reader**s map the three elements of an Internet e-mail address to the four properties as follows. The comment part of the Internet address SHOULD be ignored. **Property** names are written as "*DisplayName" to indicate that this algorithm applies to that member of any property group.

- *DisplayName: If the Internet address has a display name part, convert it to a **Unicode** string, performing decoding as specified in [RFC2047] if required, and write it to this property value. If there is no display name part, use the e-mail address part.
- *AddressType: First check whether the e-mail address was encoded by using **IMCEA** encapsulation (see section 2.1.1.8). <76> If it was, perform de-encapsulation (section 2.2.1.2) to obtain the e-mail address and type, and write the type to this property. Otherwise, write "`SMTP`" to this property value. If there is no e-mail address part, do not set this property value.
- *EmailAddress: If the Internet address was IMCEA-encapsulated, use the e-mail address obtained by de-encapsulation. Otherwise, convert the entire e-mail address part to **Unicode** and write it to this property value. If there is no e-mail address part, do not set this property value.
- *EntryId: If there is an e-mail address part, after the de-encapsulation step, perform a lookup against the address book for an entry any of whose proxy addresses matches this address. If an entry is found, construct an address book entry ID from that entry's **DN**, as specified in [MS-OXCDATA]. If no entry is found, construct a one-off entry

ID from the display name, address type, and e-mail address property values, according to the one-off entry ID specification in [MS-OXCDATA].

2.2.1.2 Recognizing and De-Encapsulating IMCEA-Encapsulated Addresses

For details about **IMCEA** encapsulation, see section 2.1.1.8. De-encapsulation SHOULD be attempted only if the domain part of the encapsulated address is recognized as local, or otherwise able to deliver mail to the de-encapsulated address. <77>

An IMCEA-encapsulated **SMTP** address consists of the following six elements:

1. The literal string "`IMCEA`" in any combination of upper or lowercase letters.
2. The original address type, one or more **ASCII** characters.
3. A literal hyphen character, U+002D.
4. The encoded original address. Legal characters are upper and lower case ASCII letters, digits, hyphen (U+002D), equal sign (U+003D), underscore (U+005F), and plus sign (U+002B). Any other characters MUST be encoded as a plus sign (U+002B) followed by two hex digits.
5. A literal "@" sign, U+0040.
6. The encapsulation domain, such as "example.com".

To identify an e-mail address as IMCEA-encapsulated, it is sufficient to match items 1-3.

To obtain the original e-mail address and type from an encapsulated address, use the following procedure:

1. Copy item 2 to the e-mail address type.
2. Extract item 4, the encoded e-mail address.
3. Decode item 4 by replacing any underscore (U+005F) with a forward slash (U+002F), and replacing any sequence of plus sign (U+002B) followed by two hex digits with the single character the hex value for which is those two digits.

2.2.1.3 From

To set the value of the **PidTagSentRepresenting property** group, **MIME** clients MUST set the From **header field** value, as specified in [RFC2822].

MIME readers MUST set the value of the **PidTagSentRepresenting property** group to the value of the first e-mail address component of the From header field (which MAY contain

multiple e-mail addresses). If the From header field contains multiple addresses, the first address MUST be used; the others MUST be ignored.

When reading **TNEF**, MIME readers SHOULD use a From header field value specified in **MIME** in preference to the **attSentFor** attribute or the **PidTagSentRepresenting** values of the **attMsgProps** attribute specified in TNEF, except for messages attached to a **TNEF message**, where a MIME header field does not exist. <78>

2.2.1.4 Sender

To set the value of the **PidTagSenderName property** group, **MIME** clients MUST set either the Sender or the From **header field** value, as specified in [RFC2822].

MIME readers MUST set the value of the **PidTagSenderName** property group to the value of the Sender header field, if the Sender header field is present in the MIME header. Otherwise, protocol servers SHOULD set the **PidTagSenderName** property group to the value of the first [RFC2822] mailbox of the From header field.

When processing **TNEF**, MIME readers SHOULD use values specified in MIME in preference to the **attFrom** attribute or the **PidTagSenderName** property group values of the **attMsgProps** attribute specified in TNEF, except for messages attached to a **TNEF message**, where a MIME header field does not exist.

2.2.1.5 To, Cc, Bcc

To set the value of a Recipient **property** group, **MIME** clients MUST set one of the To, Cc, or Bcc **header field** values, as specified in [RFC2822], that corresponds to the desired **recipient** type, as specified in the following table.

PidTagRecipientType value	Recipient type
0x00000001	To
0x00000002	Cc
0x00000003	Bcc

MIME readers MUST add one recipient to the **Message object** for each address in the To, Cc, and Bcc header fields. MIME readers MUST map the value of the Recipient property group from address elements, as specified in section 2.2.1.1. Clients MAY specify multiple To, Cc, or Bcc header fields, and **MIME readers** SHOULD process all of them.

MIME readers MUST set the value of the **PidTagRecipientType** property for each recipient row to the value specified in the table.

When processing **TNEF**, MIME readers SHOULD use values specified in MIME in preference to the value of the **attRecipTable** attribute specified in TNEF, except for TNEF DSN messages and any messages attached to a **TNEF message**.

2.2.1.6 Reply Recipients

To set the values of the **PidTagReplyRecipientEntries** and the **PidTagReplyRecipientNames** properties, **MIME** clients MUST set the Reply-To **header field** value, as specified in [RFC2822].

Note that because Reply-to is an address list and not a single address, the **property** mapping is not a normal four-property group.

MIME readers MUST set the values of the **PidTagReplyRecipientEntries** and the **PidTagReplyRecipientNames** properties (as specified in MS-OXOMSG) by mapping addresses from the Reply-To header field.

When processing **TNEF**, **MIME readers** SHOULD use a Reply-To header field value specified in MIME in preference to **PidTagReplyRecipientEntries** and the **PidTagReplyRecipientNames** values of the **attMsgProps** attribute specified in TNEF (except for messages attached to a **TNEF message**, where the MIME counterpart is not available).

2.2.1.7 Disposition Notification Recipients

To set the value of the **PidTagReadReceiptRequested** and the **PidTagReadReceipt property** group, **MIME** clients MUST set the Disposition-Notification-To **header field** value, as specified in [RFC3798].

MIME readers MUST set the value of the **PidTagReadReceiptRequested property** to TRUE if the MIME **header** contains the Disposition-Notification-To header field.

MIME readers MUST map the value of the **PidTagReadReceipt** property group from the value of the Disposition-Notification-To header field, if the field exists.

When processing **TNEF**, MIME readers SHOULD use a Disposition-Notification-To header field value specified in MIME in preference to the **PidTagReadReceiptRequested** and **PidTagReadReceipt** property group values of the **attMsgProps** attribute specified in TNEF (except for messages attached to a **TNEF message**, where the MIME counterpart is not available).

2.2.1.8 Return-Receipt-To

To set the value of the **PidTagOriginatorDeliveryReportRequested property**, **MIME** clients MUST set the [non-standard] Return-Receipt-To **header field** value.

MIME readers MUST set the value of the **PidTagOriginatorDeliveryReportRequested** property to TRUE if the message contains the Return-Receipt-To header field. The actual value of the header field is ignored, and receipts will be returned to the sender.

When processing **TNEF**, MIME readers SHOULD use a Return-Receipt-To header field value specified in MIME in preference to the **PidTagOriginatorDeliveryReportRequested** property value of the **attMsgProps** attribute specified in **TNEF** (except for messages attached to a **TNEF message**, where the MIME counterpart is not available).

2.2.2 Envelope Elements

Many **MIME header fields** that map directly to **Message object properties** have string values. Unless otherwise specified, the string values are copied directly. All string values SHOULD be converted to **Unicode** (UTF-16) before they are copied to property values, and where applicable, the decoding specified in [RFC2047] MUST be applied before generating the Unicode characters.

If there are multiple instances of a header field, **MIME reader**s SHOULD use the first instance to set the value of the corresponding property. <79> However, in the case of multiple **recipient** fields, MIME readers SHOULD combine the content of all instances to set the value of the corresponding property.

2.2.2.1 MessageID

To set the value of the **PidTagInternetMessageId property**, **MIME** clients MUST set the Message-ID **header field** value, as specified in [RFC2822]. **MIME reader**s MUST copy the value of the Message-ID header field to the **PidTagInternetMessageId** property.

2.2.2.2 Sent time

To set the value of the **PidTagClientSubmitTime property**, **MIME** clients MUST set the Date **header field** value, as specified in [RFC2822].

MIME readers MUST set the value of the **PidTagClientSubmitTime** property to the value of the Date header field, converted to **UTC**. Full precision of the Date header field, including seconds, MUST be preserved. If the Date header field is missing or contains an invalid value, MIME readers MUST set the value of the **PidTagClientSubmitTime** property to the current UTC time.

When processing **TNEF**, MIME readers MUST use a Date header field value specified in MIME in preference to an **attDateSent** or **PidTagClientSubmitTime** value specified in TNEF.

2.2.2.3 References

To set the value of the **PidTagInternetReferences property**, **MIME** clients MUST write the desired value to a References **header field**.

MIME readers MUST copy the value of the References header field to the value of the **PidTagInternetReferences** property. MIME readers MAY truncate the value of the **PidTagInternetReferences** property if it exceeds 64 KB in length.

2.2.2.4 Sensitivity

To set the value of the **PidTagSensitivity property** to a value other than normal, **MIME** clients MUST write the desired value to a Sensitivity **header field**.

MIME readers MUST map Sensitivity header field values to **PidTagSensitivity** values as specified in the following table.

PidTagSensitivity value	Sensitivity header field value
0x00000000	Normal
0x00000001	Personal
0x00000002	Private
0x00000003	Company Confidential

2.2.2.5 Importance

To set the value of the **PidTagImportance property**, **MIME** clients SHOULD write the desired value to an Importance **header field**.

MIME readers MUST map Importance header field values to **PidTagImportance** values as specified in the following table.

Importance header field value	**PidTagImportance** value

Importance header field value	**PidTagImportance** value
Low	0x00000000
Normal	0x00000001
High	0x00000002

MIME clients MAY use a Priority, X-Priority, or X-MSMail-Priority header field instead of an Importance header field to set the value of the **PidTagImportance** property. In that case, MIME readers MUST map the header field values to **PidTagImportance** values, as specified in the following tables. However, if an Importance header field is present, MIME readers SHOULD use its value in preference to any of the others.

Priority header field value	**PidTagImportance** value
NonUrgent	0x00000000
Normal	0x00000001
Urgent	0x00000002

X-Priority header field value	**PidTagImportance** value
5	0x00000000
4	0x00000000
3	0x00000001
2	0x00000002
1	0x00000002

X-MSMail-Priority header field value	**PidTagImportance** value
Low	0x00000000
Normal	0x00000001
High	0x00000002

2.2.2.6 Subject

To set the value of the **PidTagSubjectPrefix** and **PidTagNormalizedSubject properties**, **MIME** clients MUST set the Subject **header field** value, as specified in [RFC2822].

MIME readers SHOULD analyze the Subject header field value into a prefix and a normalized subject value, as specified in section 2.2.2.6.1, and then set the values of the **PidTagSubjectPrefix** and **PidTagNormalizedSubject** properties, rather than simply setting the value of **PidTagSubject**. MIME readers MAY truncate the Subject value; the first 255 characters is a typical length restriction.

MIME readers MUST use a Subject header field value specified in **MIME** in preference to an **attSubject** or **PidTagSubject** value specified in **TNEF**. They SHOULD, however, use **PidTagSubjectPrefix** and **PidTagNormalizedSubject** values from TNEF when they match the MIME subject, because of limitations in the subject normalization algorithm of section 2.2.2.6.1.

2.2.2.6.1 Normalizing the Subject

If no values are available for **PidTagNormalizedSubject** and **PidTagSubjectPrefix** in the **MIME** message, protocol servers SHOULD parse the Subject **property** value and set those values as follows. If the Subject **header field** value consists of one, two, or three characters (exclusive of colon (U+003A), blank (U+0020), or digits (U+0030 through U+0039)), followed by a colon (U+003A) and any number of blanks (U+0020), the protocol server SHOULD set the value of **PidTagSubjectPrefix** to the aforementioned one, two, or three characters appended with a colon and a space (": "), and SHOULD set the value of **PidTagNormalizedSubject** to the remainder of the Subject header field value beginning immediately after the aforementioned blanks.

2.2.2.7 Conversation Topic

To set the value of the **PidTagConversationTopic property**, **MIME** clients MUST write the desired value to a Thread-Topic **header field**. This value SHOULD be the same as the value of the Subject header field, normalized as specified in section 2.2.2.6.1 to remove any prefix.

MIME readers MUST copy the value of a Thread-Topic header field to the value of the **PidTagConversationTopic** property. MIME readers SHOULD also use this header field value as a hint to normalize the subject, as specified in section 2.2.2.6.1, if this value matches the tail of the Subject header field value.

2.2.2.8 Conversation Index

To set the value of the **PidTagConversationIndex property**, **MIME** clients MUST write the desired value to a Thread-Index **header field**. The property data type is binary, and protocol clients MUST encode the header field value using base64 encoding, as specified in [RFC2045]. The format of the desired value is specified in [MS-OXOMSG].

MIME readers MUST copy the value of a Thread-Index header field to the value of the **PidTagConversationTopic** property, assuming the base64-encoded text can be successfully decoded to binary data. MIME readers SHOULD ignore a Thread-Index **header** that does not contain base64-encoded binary data.

2.2.2.9 In-Reply-To Message ID

To set the value of the **PidTagInReplyToId property**, **MIME** clients MUST write the desired value to an In-Reply-To **header field**, as specified in [RFC2822].

MIME readers MUST copy the value of an In-Reply-To header field to the value of the **PidTagInReplyToId** property.

2.2.2.10 ReplyBy Time

To set the value of the **PidTagReplyTime property**, **MIME** clients MUST set the Reply-By **header field** value, as specified in [RFC2822].

MIME readers MUST set the value of the **PidTagReplyTime property** to the value of the Date header field, converted to **UTC** time.

When processing **TNEF**, MIME readers MUST use a Reply-By header field value specified in MIME in preference to a **PidTagReplyTime** value specified in TNEF.

2.2.2.11 Language Properties

To set the value of the **PidTagMessageLocaleId property**, **MIME** clients MUST set the Content-Language header, as specified in [RFC3282].

MIME readers MUST set the value of the **PidTagMessageLocaleId** property by extracting the first language tag from the value of the Content-Language header and mapping it to an LCID, as specified in [MS-LCID]. MIME readers SHOULD use the value of a Content-Language **header field** in preference to the value of **PidTagMessageLocaleId** found in the **attMsgProps** attribute of a **TNEF message**.

To set the value of the **PidNameAcceptLanguage** property, MIME clients SHOULD write an Accept-Language header field with the desired value. MIME clients MAY write an X-Accept-Language header field instead.

MIME readers SHOULD copy the value of either header field to the value of the **PidNameAcceptLanguage** property. If both header fields are present, MIME readers SHOULD use the Accept-Language header field. <80><81>

2.2.2.12 Categories

To set the value of the **PidNameKeywords property**, **MIME** clients MUST set the Keywords **header field**, as specified in [RFC2076].

MIME readers SHOULD map the value of a Keywords header field to the value of the **PidNameKeywords property** by splitting the Keywords header field value at each comma (U+0032), trimming whitespace, and storing each keyword as an individual value of the multiple string property.

To prevent conflicts among category schemes in different organizations, MIME readers MAY omit mapping the Keywords header field to the **PidNameKeywords** property.

2.2.2.13 Message Expiry Time

To set the value of the **PidTagExpiryTime property**, **MIME** clients MUST set the Expires **header field** to the desired value.

MIME readers MUST copy the value of the Expires header field to the value of the **PidTagExpiryTime** property, after converting it to **UTC** time.

MIME clients MAY use an Expiry-Date header field instead of an Expires header field. Protocol servers MUST use the value of the Expires heapeanut der field in preference to Expiry-Date, if both header fields are present.<82>

2.2.2.14 Suppression of Automatic Replies

To set the value of the **PidTagAutoResponseSuppress property** to -1, indicating that all automatic replies to the message are to be suppressed, MIME clients SHOULD write an X-AUTO-Response-Suppress **header field** with the value "All". **MIME** clients MAY, instead, write a Precedence **header** field with any value.

To set the value of the **PidTagAutoResponseSuppress** property to a more specific value, MIME clients MUST write an X-AUTO-Response-Suppress header field with one or more values from the table in section 2.1.2.20 selected.

MIME readers SHOULD map individual elements of an X-Auto-Response-Suppress header field to bits in the value of the **PidTagAutoResponseSuppress** property according to the table. If both X-Auto-ResponseSuppress and Precedence header fields are present, the **PidTagAutoResponseSuppress** property value SHOULD be 0xFFFFFFFF. If the value of the X-Auto-Response-Suppress header field is other than as specified in the table in section 2.1.2.20, MIME readers SHOULD ignore the entire header field. <83><84>

2.2.2.15 Content Class

To set the value of the **PidNameContentClass property**, **MIME** clients MUST write a Content-Class **header field** with the desired value.<85>

MIME readers MUST copy the value of a Content-Class header field to the value of the **PidNameContentClass** property.

MIME readers SHOULD also set the value of the **PidTagMessageClass** property for certain Content-Class header field values as specified in the following table, but only if the value of **PidTagMessageClass** would otherwise be set to "`IPM.Note`".

Content-Class header field value	**PidTagMessageClass** property value
"fax"	"IPM.Note.Microsoft.Fax"
"fax-ca"	"IPM.Note.Microsoft.Fax.CA"
"missedcall""	"IPM.Note.Microsoft.Missed.Voice"
"voice-uc"	"IPM.Note.Microsoft.Conversation.Voice"
"voice-ca"	"IPM.Note.Microsoft.Voicemail.UM.CA"
"voice"	"IPM.Note.Microsoft.Voicemail.UM"

Content-Class header field value	**PidTagMessageClass** property value
Starts with "urn:content-class:custom."	"IPM.Note.Custom.", followed by the value of Content-Class header field, with "urn:content-class:custom." prefix removed. <86><87>

Additionally, if the Content-Class **header field** value begins with "`InfoPath.`", then **MIME readers** SHOULD extract a substring from the header field value beginning immediately after the prefix. If this string contains a period character (U+002E), and the first occurrence of this character is not the last one in the string, this string SHOULD be further separated into two substrings. The delimiting period is not included into either one of the substrings.

The first substring SHOULD be additionally checked to match the string format of a **GUID** string (see [MS-DTYP]). If this check succeeds, the second substring SHOULD be saved as a value of the **PidLidInfoPathFormName** property. In addition, the first substring SHOULD be appended to "`IPM.InfoPathForm.`" and written to the value of the **PidTagMessageClass** property.

If a message that is being processed by a MIME reader is clear signed or opaque signed, as specified in [MS-OXOSMIME], the appropriate suffix ("`.SMIME.MultipartSigned`" or "`.SMIME`") SHOULD be appended to the value of **PidTagMessageClass**.

2.2.2.16 Message Flagging

To set the value of the **PidLidFlagRequest property**, **MIME** clients MUST write an X-Message-Flag header with the desired value.

MIME readers MUST copy the value of an X-Message-Flag **header** to the value of the **PidLidFlagRequest** property. In addition, when an X-Message-Flag header is present, MIME readers SHOULD do all the following:

1. Set the value of the **PidTagFlagStatus** property to 2 (denoting that the message is flagged).
2. Copy the value of the **PidTagSubject** property to the value of the **PidLidToDoTitle** property.
3. Set the value of the **PidLidTaskStatus** property to 0 (zero) (denoting that a task is not started).
4. Delete or disregard any existing property values for the following properties:

PidLidTaskDueDate
PidLidTaskStartDate
PidTagFlagCompleteTime
PidLidTaskDateCompleted

5. Set the value of the **PidLidTaskComplete** property to FALSE.
6. Set the value of the **PidLidPercentComplete** property to 0.0.
7. Set the value of the **PidTagToDoItemFlags** property to 8.

2.2.2.17 List Server Properties

To set the values of list server–related **properties**, MIME clients MUST write **header field**s as specified in the following table.

Property	Preferred header field name	Alternate header field name
PidTagListHelp	List-Help	X-List-Help
PidTagListSubscribe	List-Subscribe	X-List-Subscribe
PidTagListUnsubscribe	List-Unsubscribe	X-List-Unsubscribe

MIME readers MUST copy header field values to property values as specified in the table.

2.2.2.18 Payload Properties

To set the value of the **PidTagAttachPayloadClass** or **PidTagAttachPayloadProviderGuidString properties**, **MIME** clients SHOULD write an X-Payload-Class and an X-Payload-Provider-Guid **header field**, respectively. Such header fields SHOULD be written to a **MIME entity** that will be analyzed as an attachment, as specified in section 2.2.4.<88>

MIME readers MUST copy these header field values to the values of the corresponding properties. MIME readers SHOULD ignore these header fields when they appear on a MIME entity that is analyzed as a message or **message body**, rather than as an attachment.

2.2.2.19 Classification Properties

To set **property** values related to message classification, **MIME** clients SHOULD write the following **header field**:

```
X-Microsoft-Classified: true
```

In addition, MIME clients SHOULD write header field values for all of X-Microsoft-Classification, X-Microsoft-ClassDesc, X-Microsoft-Classification-Guid, and X-Microsoft-Classification-Keep.

When the appropriate X-Microsoft-Classified header field is present, **MIME reader**s SHOULD map or copy all classification header field values to their corresponding property values, as specified in the following table. If the X-Microsoft-Classified header field is missing or has a different value, MIME readers SHOULD NOT set any of the five property values listed in the table. <89><90>

Classification header field	Classification property	Header value mapping
X-Microsoft-Classified	**PidLidClassified**	"true" => true
X-Microsoft-ClassKeep	**PidLidClassificationKeep**	"true" => true "false" => false
X-Microsoft-Classification	**PidLidClassification**	copy string value
X-Microsoft-ClassDesc	**PidLidClassificationDescription**	copy string value
X-Microsoft-ClassID	**PidLidClassificationGuid**	copy string value

2.2.2.20 Unified Messaging Properties

To set the values of unified messaging properties, **MIME** clients SHOULD write the desired value to the corresponding **header field**, as specified in the following table.

Header field name	Property
X-CallingTelephoneNumber	**PidTagSenderTelephoneNumber**
X-VoiceMessageSenderName	**PidTagVoiceMessageSenderName**
X-AttachmentOrder	**PidTagVoiceMessageAttachmentOrder**

Header field name	Property
X-CallID	**PidTagCallId**
X-VoiceMessageDuration	**PidTagVoiceMessageDuration**; header value MUST be parsed as **PtypInteger32**
X-FaxNumberOfPages	**PidTagFaxNumberOfPages**; header value MUST be parsed as **PtypInteger32**

MIME readers SHOULD copy header field values to **property** values, as specified in the table. <91><92>

2.2.2.21 Content-ID

To set the value of the **PidTagBodyContentId property**, **MIME** clients MUST write the desired value to a Content-ID **header field** on a **MIME entity** that maps to a **message body**, as specified in section 2.2.3.

MIME readers SHOULD copy the value of a Content-ID header field on such a MIME entity to the value of the **PidTagBodyContentId** property. <93><94>

MIME clients MAY write either a Content-ID or a Content-Location header field, but SHOULD NOT write both on a single MIME entity.

2.2.2.22 Content-Base

To set the value of the **PidNameContentBase property**, **MIME** clients MUST write the desired value to a Content-Base **header field** on a **MIME entity** that maps to a **message body**, as specified in section 2.2.3.

MIME readers MUST copy the value of a Content-Base header field on such a MIME entity to the value of the **PidNameContentBase** property.

To set the value of the **PidNameContentBase** property, MIME clients SHOULD write the desired value to a Content-Base header field on a MIME entity that maps to a **Message object** (top-level or attached).

MIME readers SHOULD copy the value of a Content-Base header field on such a MIME entity to the value of the **PidNameContentBase** property. <95>

2.2.2.23 Content-Location

To set the value of the **PidTagBodyContentLocation property**, MIME clients SHOULD write the desired value to a Content-Location **header field** on a **MIME entity** that maps to a **message body**, as specified in section 2.2.3.

MIME readers SHOULD copy the value of a Content-Location header field on such a MIME entity to the value of the **PidTagBodyContentLocation** property. <96>

2.2.2.24 XRef

To set the value of the **PidNameCrossReference property**, **MIME** clients MUST write the desired value to an XRef **header field**.

MIME readers MUST copy the value of an XRef header field to the value of the **PidNameCrossReference** property.

2.2.2.25 PidTagTransportMessageHeaders

MIME readers SHOULD copy all **header fields**, with certain exceptions, from an inbound message to the value of the **PidTagTransportMessageHeaders property**. With the exception of **headers** specifically mentioned in section 2.1.2, headers that begin with the reserved name prefixes "X-MS-Exchange-Organization-" and "X-MS-Exchange-Forest-" SHOULD NOT be copied to **PidTagTransportMessageHeaders**.

Clients SHOULD NOT set the value of **PidTagTransportMessageHeaders**. This property value SHOULD be set only by MIME readers upon delivery of a message from **SMTP**, in which case it SHOULD be set to the header of the top-level message (with exceptions as already specified).

2.2.2.26 Generic Header Fields in PS_INTERNET_HEADERS

To create a named **property** in the PS_INTERNET_HEADERS property set, whose name is a **header field** name and whose value is a header field value, **MIME** clients MUST create a header field with the desired name and value.

For each such header field, **MIME reader**s SHOULD create a named property as follows: <97><98>

- The PropertyName **GUID** is "%X86.03.02.00.00.00.00.00.00.C0.00.00.00.00.00.46".
- The PropertyName name is the header field name.
- The property value is the header field value. If the header field value was encoded according to [RFC2047], MIME readers MUST decode it.

MIME readers MUST NOT create such named properties for any MIME header field that is mapped to a different property, as specified elsewhere in this section. MIME readers SHOULD NOT create such named properties for any of the following MIME header fields:

- Received
- Resent-From
- Resent-Sender
- Resent-Date
- Resent-Message-Id
- Content-Type
- Content-Disposition
- Content-Description
- Content-Transfer-Encoding
- Content-ID
- Content-MD5
- Mime-Version
- Return-Path
- Comments
- AdHoc
- Apparently-To
- Approved
- Control
- Distribution
- Encoding
- FollowUp-To
- Lines
- Bytes
- Article
- Supercedes
- NewsGroups
- NntpPostingHost
- Organization
- Path
- RR
- Summary
- Trace
- Encrypted
- X-MimeOle
- X-MS-Tnef-Correlator

- Any header field the name of which begins with "X-MS-Exchange-Organization-" or "X-MS-Exchange-Forest-", except for the following:
    - X-MS-Exchange-Organization-AuthAs
    - X-MS-Exchange-Organization-AuthDomain
    - X-MS-Exchange-Organization-AuthMechanism
    - X-MS-Exchange-Organization-AuthSource

2.2.3 Body Text

Unlike **MIME**, which allows an arbitrary number of inline text **body parts**, **Message objects** distinguish one text body part as the **message body**.

2.2.3.1 Client Actions

To send the value of **PidTagBody** as the definitive body text, **MIME** clients SHOULD create a MIME message in which the first or only element has "text/plain" as the value of the Content-Type **header field**, and that element's body contains the text. MIME clients SHOULD specify the **charset** of the **message body** text on the corresponding MIME element.

To send the value of **PidTagHtml** as the definitive body text, MIME clients SHOULD create a MIME message in which the first or only MIME element has "text/html" as the value of the Content-Type header field, and that element's body contains a well-formed HTML document. Clients SHOULD generate a multipart/alternative structure with a text/plain representation (see Example 1), so that a greater number of clients can process the message.

To send the value of **PidTagRtfCompressed** as the definitive body text, MIME clients SHOULD create a MIME message that contains a **TNEF body part**, as specified at the beginning of section 2, and write the desired value of **PidTagRtfCompressed** into the **attMsgProps** attribute of the TNEF.

2.2.3.2 Determining Which MIME Element Is the Message Body

The rules a **MIME reader** MUST follow for selecting a **message body** are both qualifying, or positive, and disqualifying, or negative. To qualify as a message body, a **MIME entity** MUST meet at least one of the following conditions:

- Content-Type **header field** value is "text/plain", "text/html", "text/enriched", or "text/calendar".
- Content-Type header field value is "multipart/alternative" and at least one child MIME entity is "text/plain", "text/html", "text/enriched", or "text/calendar".

- Content-Type header field value is "`multipart/related`", and its first child MIME entity is either "`text/html`", or "`multipart/alternative`" with at least one text/html child MIME entity.

To qualify as a message body, a MIME entity MUST NOT have a Content-Disposition header field with the value "`Attachment`".

In all cases, it is the text **body part** and not the containing multipart itself that is mapped to the message body.

MIME readers MUST select the first MIME entity that qualifies according to the rules as the message body. MIME readers SHOULD then map the content of the selected MIME entity to a **Message object property** value according to the following rules:

- If the body MIME entity is a single text/plain, copy its content to the value of the **PidTagBody** property.
- If the body MIME entity is a single text/html, copy its content to the value of the **PidTagHtml** property.
- If the body MIME entity is a single text/enriched, convert its content to **HTML** and copy the result to the value of the **PidTagHtml** property.
- If the body MIME entity is a single text/calendar, parse the iCalendar document and copy the value of the DESCRIPTION property to **PidTagBody**. If the DESCRIPTION property is missing, MIME readers MAY use the value of the COMMENT property instead. For details, see [MS-OXCICAL].
- If the body MIME entity is multipart/alternative, MIME readers SHOULD select the last child entity that has one of the four eligible types and map it. However, if the last child entity is text/calendar and one of the preceding entities is text/html, MIME readers MAY map the text/html instead of the DESCRIPTION property of the text/calendar.
- If the body MIME entity is multipart/related, identify the first child MIME entity that is either text/html or multipart/alternative and map it according to rules 1-5.

2.2.3.2.1 Selecting the Primary Message Text MIME Element

When alternative text **MIME** elements are present and eligible for use as the **message body**, as specified in section 2.2.3.2, **MIME reader**s SHOULD choose a MIME element to populate the message body text by using the following ranking of content types:

- text/html
- text/enriched

- text/plain
- text/calendar (but only if the METHOD **property** value of the text/calendar **body part** is PUBLISH, REQUEST, REPLY, or CANCEL)

If text/html is selected, MIME readers MUST copy the MIME element body text to the value of the **PidTagHtml property**, map the *charset* parameter of the MIME element's Content-Type **header field** to a **code page**, and set the value of the **PidTagInternetCodepage** property to that code page. If the *charset* parameter is not present, MIME readers MAY use the value of a Content-Type meta tag in the **HTML** document, but SHOULD verify its validity before using it.

If text/plain is selected, MIME readers MUST convert the **plain text** to UTF-16LE and write the resulting text to the value of the **PidTagBody** property. MIME readers SHOULD, in addition, map the value of the *charset* parameter of the MIME element's Content-Type header field to a code page, and set the value of the **PidTagInternetCodepage** property to that code page.

If text/enriched is selected, **MIME readers** MUST convert to either text/plain, text/html, or **RTF**, and handle that as previously specified.

If both text/html and text/calendar body parts are present and eligible for use as message body, instead of writing text to the **PidTagHtml** property, MIME readers SHOULD convert the HTML text to RTF and write it to the value of the **PidTagRtfCompressed** property. Alternatively, MIME readers MAY choose to use plain text from a text/plain body part or from data in the text/calendar body part, as specified in [MS-OXCICAL]. MIME readers MUST NOT set the **PidTagHtml** property on a calendar or meeting **Message object**.

2.2.4 Attachments

During **MIME analysis**, **MIME reader**s SHOULD classify all non-multipart **MIME entities** and multipart/appledouble MIME entities (that contain appropriate child MIME sub-parts) into the following three categories:

1. MIME entities that can potentially represent the **message body**.
2. MIME entities that represent non-inline attachments.
3. MIME entities that represent attachments that can potentially be inline.

All MIME entities that can be classified as attachments (2nd or 3rd category) SHOULD be treated by MIME readers as attachment MIME parts, and an entry in an attachment table SHOULD be created for each such MIME part. However, depending on the value of the Content-Type MIME header, analysis SHOULD be done differently, as follows:

a) Message/rfc822 MIME entities SHOULD be treated as embedded message attachments, as specified in section 2.2.4.3.

b) Multipart/appledouble, application/applefile, and application/mac-binhex40 MIME entities SHOULD be treated as Macintosh attachments, as specified in section 2.2.4.2.

c) Message/external-body attachments SHOULD be treated as external body attachments, as specified in section 2.2.5.

d) All other attachments SHOULD be treated as regular file attachments, as specified in section 2.2.4.1.

If no Content-Type **header field** is present on a MIME entity, MIME readers SHOULD treat it as text/plain (unless this MIME entity is a sub-part of multipart/digest, in which case the default value for the Content-Type MIME header field is message/rfc822).

2.2.4.1 Regular File Attachment MIME Part Analysis

When creating an **Attachment object** for a regular file attachment, **MIME reader**s MUST set the value of the **PidTagAttachMethod property** to 0x00000001.

2.2.4.1.1 File name

The attachment file name SHOULD be determined by **MIME reader**s in the following order:

1. If the Content-Disposition **header field** exists on the attachment **MIME entity**, and a non-empty *filename* parameter is available on this header, the *filename* parameter value SHOULD be used, else

2. If the Content-Type header field is available on the attachment MIME entity, and a non-empty *name* parameter is available on this **header**, the *name* parameter value SHOULD be used, else

3. If the Content-Transfer-Encoding header field is set to "`binhex`", MIME readers SHOULD try to parse MIME part body as **MacBinary** structure, as specified in section 2.2.4.2.3. Only the first 128 bytes of the **MIME body** (decoded with binhex, see [RFC1741]) SHOULD be parsed. If parsing of **MacBinary** structure succeeds, file name data from this structure SHOULD be used, else

4. If the attachment MIME part body is encoded with uuencode (see section 2.3.1), and it contains file name data, this file name SHOULD be used, else

5. If the Content-Description header field is available on the attachment and its value is non-empty, it SHOULD be used as the file name value for an attachment. (Even if a file name for an attachment was found in one of the previous steps, this value SHOULD be written to **PidTagDisplayName** for an attachment.)

MIME readers SHOULD sanitize the resulting file name and display name by removing characters that are not legal in a Windows file name. Invalid characters are listed in the following table.

Description	Code point	Character
Control characters	U+0000 through U+001F	
Double quote	U+0022	"
Forward slash	U+002F	/
Colon	U+003A	:
Left angle bracket	U+003C	<
Right angle bracket	U+003E	>
Pipe	U+0049	|
Backslash	U+005C	\

The following steps SHOULD then be applied both to the attachment file name and the display name (if the display name is not available, the empty string SHOULD be used):

- Replace all **Unicode** separator characters with spaces.
- Separate name into base and extension parts. The extension is defined as the trailing part of a name that starts after the last appearance of a '.' character (U+002E) in the name, or an empty string if name contains no such character.
- Remove all leading and trailing spaces and leading and trailing '.' (U+002E) characters from both the base and the extension.

If the extension part of the display name is not empty and does not match the extension part of file name, it SHOULD be appended to the base part of display name.

If the file name base and/or file name extension is empty, the **MIME reader** SHOULD generate some non-empty strings for the attachment file name base and/or extension.

After that, if the base part of the display name is empty, it SHOULD be replaced with the base part of the file name. Finally, the file name base, file extension, and display name SHOULD be reassembled from the base and extension parts and saved in the appropriate properties, as specified in the following table.

Property	Value
PidTagDisplayName	<display name base>.<file name extension>
PidTagAttachLongFilename	<file name base>.<file name extension>
PidTagAttachExtension	.<file name extension>

The value saved to **PidTagAttachLongFilename** SHOULD be further processed to form a valid 8.3 file name, and then written to **PidTagAttachFilename**, as follows:

1. The value SHOULD be first separated into name and extension parts, using the last '.' character (U+002E) as a separator. If no such character is present, or the only appearance of this character is in the beginning of the file name, the extension is considered to be empty; the separator character itself is not included into the name or extension.
2. Replace the following characters with an underscore (U+005F):
plus sign "+" (U+002B), comma ',' (U+002C), equal sign '=' (U+003D), left square bracket '[' (U+005B), right square bracket ']' (U+005D), semicolon ';' (U+003B).
3. Remove the following characters: space (U+0020), period '.' (U+002E), apostrophe '\" (U+0027), asterisk '*' (U+002A), question mark '?' (U+003F), as well as characters with a UTF8 code greater than 127.
4. If name is empty after removing such characters, **MIME readers** SHOULD generate a non-empty value.
5. Trim the name part of the file name to 8 characters, and the extension part to 3 characters.
6. If either name or extension was shortened, the name part SHOULD additionally be trimmed to 6 characters, and "~1" SHOULD be added to its end.
7. Recombine the file name and extension, separated by a single '.' (U+002E).

2.2.4.1.2 Content Type

MIME readers SHOULD save the value of the Content-Type **MIME header field** in the **PidTagAttachMimeTag property** during **MIME analysis**. The following notes apply for specific values of this header field:

- The "application/ms-tnef" value SHOULD be replaced with "application/octet-stream". This is in the rare case when a TNEF body part is corrupt and cannot be completely processed. Ordinarily, a TNEF body part SHOULD NOT be written to an attachment, but analyzed into Message object properties and discarded.
- The values "`application/x-pkcs7-mime`" and "`application/pkcs7-mime`": the entire Content-Type header field value, including all parameter names and values, SHOULD be written to the **PidNameContentType** property value.
- For Content-Type values that start with "`text/`", if a *charset* parameter is present, the parameter value SHOULD be written to the **PidTagTextAttachmentCharset** property.

2.2.4.1.3 Attachment Creation and Modification Date

If a Content-Disposition **MIME header field** is present on the attachment MIME part, **MIME reader**s SHOULD use its parameters to set creation and modification dates on the **Attachment object**. <99> If a parameter is missing or its value is not a valid date, the corresponding **property** value SHOULD NOT be set. Date and time values MUST be translated to **UTC**.

Content-Disposition parameter name	Property
creation-date	**PidTagCreationTime**
modification-date	**PidTagLastModificationTime**

2.2.4.1.4 Attachment Content-Id, Content-Base, and Content-Location

If a Content-Id **MIME header** is present on the attachment MIME part, **MIME reader**s SHOULD copy its value to the **PidTagAttachContentId property**. If this value starts with '<' (U+003C) and/or ends with '>' (U+003E), these characters SHOULD be removed.

If a Content-Location MIME header is present on the attachment MIME part, its value SHOULD be saved in the **PidTagAttachContentLocation** property.

If a Content-Base MIME header is present on the attachment MIME part, its MIME readers SHOULD copy its value to the **PidTagAttachContentBase** property.

Additionally, if an attachment MIME part is a child of a multipart/related MIME element, and either a Content-Id or Content-Location **MIME header** is present, MIME readers SHOULD mark the message as inline, as specified in section 2.1.4.1.2. MIME readers SHOULD verify whether the attachment is actually referenced from the **message body**, and mark it as inline only if that is the case; but MAY mark it as inline unconditionally.<100>

2.2.4.1.5 Attachment Content-Transfer-Encoding and MIME Part Body

As specified in [RFC2045], a Content-Transfer-Encoding **header** might be present on the attachment **MIME** part. **MIME reader**s SHOULD support the following values for this header:

- Base64. See [RFC2045], [RFC4648].
- Quoted-printable. See [RFC2045]
- 7 bit. See [RFC2045]
- 8 bit. See [RFC3516]
- Binary. See [RFC3030]
- Mac-binhex40
- X-uuencode
- X-uue

As specified in [RFC2045], if the Content-Transfer-Encoding MIME header is missing, MIME readers MUST behave as if it were set to 7 bit.

The attachment's content SHOULD be decoded by using the appropriate decoding procedure and saved as the value of the **PidTagAttachDataBinary property**. MIME readers SHOULD, as a rule, use **RopOpenStream** (as specified in [MS-OXCROPS]) to create this property value.

The encoding values mac-binhex40, x-uuencode, and x-uue are non-standard. If a "mac-binhex40" Content-Transfer-Encoding value is encountered, MIME readers SHOULD treat the MIME part body as if it had a Content-Type **header field** value of "application/mac-binhex40" and process it as specified in section 2.2.4.2.3.

However, in the unlikely case of an actual "application/mac-binhex40" Content-Type, MIME readers SHOULD extract only the data fork from the MIME element content and use it as the value of the **Attachment object's PidTagAttachDataBinary** property. For X-uuencode and X-uue values, MIME readers SHOULD treat the attachment content as encoded with uuencode (see [IEEE1003.1]). The decoded value SHOULD be written to the Attachment object's **PidTagAttachDataBinary** property value.

2.2.4.1.6 AttachmentContent-ID, Content-Base, and Content-Location

To set the value of the **PidTagAttachContentId property**, **MIME** clients MUST write the desired value to a Content-ID **header field** on a **MIME entity** that maps to an **Attachment object**, as specified in section 2.2.4.

MIME readers MUST copy the value of a Content-ID header field on such a MIME entity to the value of the **PidTagAttachContentId** property.

To set the value of the **PidTagAttachContentBase** property, MIME clients MUST write the desired value to a Content-Base header field on a MIME entity that maps to an AAttachment object, as specified in section 2.2.4.

MIME readers MUST copy the value of a Content-Base header field on such a MIME entity to the value of the **PidTagAttachContentBase** property.

To set the value of the **PidTagAttachContentLocation** property, MIME clients MUST write the desired value to a Content-Location header field on a MIME entity that maps to an Attachment object, as specified in section 2.2.4.

MIME readers MUST copy the value of a Content-Location header field on such a MIME entity to the value of the **PidTagAttachContentLocation** property.

2.2.4.2 Apple File Formats

[RFC1740] and [RFC1741] specify the use of the **MIME** Content-Types multipart/appledouble, application/applefile, and application/mac-binhex40 to encode files that originate from a Macintosh operating system, to preserve additional data that might be available for these files in that operating system. **MIME reader**s SHOULD preserve this additional data for attached files to enable full support of Macintosh-based client applications.

In particular, the **Attachment object** content that is stored in **PidTagAttachDataBinary** MUST contain a MacBinary stream, as specified in [MacBin]. This stream format incorporates both the resource and data forks, as well as certain metadata.

Note that **MIME analysis** of application/applefile attachments is specified differently, depending on whether the application/applefile **MIME entity** is a sub-part of multipart/appledouble.

2.2.4.2.1 Multipart/Appledouble

A **MIME** element with Content-Type multipart/appledouble **MIME readers** MUST, as specified in [RFC1740], have two child MIME elements. The "header part" MUST have a Content-Type of application/applefile; the "data part" MAY have any MIME Content-Type except application/applefile or another multipart Content-Type.

As a MIME reader copies data from a multipart/appledouble **MIME entity** to an **Attachment object**, it MUST analyze the three parts in the following sequence:

1. The data part (typically the first child of multipart/appledouble).
2. The header part (typically the second child of multipart/appledouble).
3. The multipart/appledouble envelope itself.

Property valuesthat are set as a result of MIME header analysis of a particular MIME part MUST overwrite property values that are set as a result of previous MIME part analysis.

The procedure of **header field** analysis for any part of a multipart/appledouble MIME part is similar to the procedure for ordinary file attachments specified in section 2.2.4.1, with the following additions:

1. MIME readers MUST set the value of the **PidTagAttachMimeTag property** to "`multipart/appledouble`".
2. MIME readers MUST set the value of the **PidTagAttachEncoding** property to the following byte sequence (in hexadecimal): "%x2A.86.48.86.F7.14.03.0B.01".
3. MIME readers MUST copy the value of the Content-Type header field on the data part to the value of the **PidNameAttachmentMacContentType** property.
4. MIME readers SHOULD copy the entire **MIME body** of the header part to the value of the Attachment object's **PidNameAttachmentMacInfo** property. MIME readers SHOULD also parse this data as an **AppleSingle** structure, as specified in [RFC1740], and combine it with the MIME body from the data part to form a **MacBinary** structure, which SHOULD then be written to the Attachment object's **PidTagAttachDataBinary** property.
5. MIME readers MUST copy file creator and file type information taken from the MacBinary representation of the attachment, to the value of the **PidTagAttachAdditionalInformation** property, with special formatting as follows; the file creator and type fields are both unsigned 32-bit integers in **big-endian** format:
    a. A single byte, value "0x2E" (colon character).
    b. The file creator, encoded by the rule that follows.
    c. A single byte, value "0x2E" (colon character).
    d. The file type, encoded by the rule that follows.
    e. A single byte, value "0x00".
    f. Encoding is done from the highest-order byte to the lowest-order byte, by using the following scheme:

- Single bytes with values for '\' (%x5C), ':' (%x3A), and ';' (%x3B) are replaced with two-byte sequences: '\\' (%x5C.5C), '\:' (%x5C3A), and '\;' (%x%c3B) respectively.
- Single bytes with values less than 32, greater than 251, or equal to 127 are encoded by a backslash (%x5C), followed by the byte value in octal, padded with zeroes to 3 digits. So, for example, a "0x01" byte is encoded as '\001', and "0xFF" is encoded as '\377'.

If parsing of the header part fails, MIME readers SHOULD reject the entire message as not MIME compliant.

If the **AppleSingle** structure from the header part contains a file name for this attachment, it SHOULD be used as the file name only if no file name was found during processing of the MIME **headers**. <101>

2.2.4.2.2 Application/Applefile

This section specifies **MIME analysis** for **MIME** parts with Content-Type application/applefile which are not sub-parts of a MIME part with Content-Type multipart/appledouble.

The procedure of MIME **header** analysis for application/applefile attachments is the same as for the procedure for ordinary file attachments specified in section 2.2.4.1, with one exception:

- The **MIME reader** MUST set the value of the **PidTagAttachMimeTag property** to "`application/applefile`".

Processing of MIME content SHOULD include parsing the **AppleSingle** structure, defined in [RFC1740]. MIME readers SHOULD use the data from this structure to fill the **PidTagAttachDataBinary** property and the **PidNameAttachmentMacInfo** property with appropriate structures, as specified in section 2.2.4.2.1.

If **MIME body** data does not match the definition of the **AppleSingle** structure (see [RFC1740]), MIME readers MAY choose to try to interpret the body of this MIME part as a **MacBinary** structure. If this succeeds, MIME readers SHOULD copy the resulting **MacBinary** structure to the value of the **PidTagAttachDataBinary** property, and **PidNameAttachmentMacInfo** SHOULD be filled with appropriate data from the **MacBinary** structure. The value of the **PidNameAttachmentMacInfo** property SHOULD be application/applefile data that contains only the header and resource fork sections. But if the MIME reader fails to parse the MIME body, the entire message SHOULD be rejected as not MIME compliant.

If the **AppleSingle** or **MacBinary** structure contains a file name for this attachment, it SHOULD be used only if no file name was found during analysis of the attachment's MIME headers. <102>

The remainder of this section specifies how MIME readers SHOULD map elements from AppleSingle format, which can have Content-Type of multipart/appledouble, or application/applefile, to MacBinary data in the value of the **PidTagAttachDataBinary** property.

The general structure of AppleSingle format is described in [RFC1740]. Is short, this data structure contains a header part, followed by some number of entries. Each of these entries is identified by a number (**AppleSingleEntryId**, unsigned 32 bit integer), which defines the internal structure of its binary data. The value of each **AppleSingleEntryId**, along with the definition of the structure of each entry, is specified by [RFC1740]. Custom entries are also allowed in this format.

The **MacBinary** structure is described in [MacBin]. It consists of the following five parts; each part MUST be padded to a 128 byte boundary, and all parts except the header are optional: <103>

1. Header
2. Additional header data
3. Actual file data (data fork)
4. Resource fork
5. Comment

The structure of the **MacBinary** header, with comments on usage of each field by MIME readers, is shown in the following table. All offsets and lengths are in bytes, and all integers use **big-endian** byte ordering.

Field offset	Field length	Description
0	1	Old version number, MUST be zero.
1	1	Length of file name, unsigned byte; MUST be less than 64.
2	63	File name, in **ASCII**; characters beyond the length specified in byte 1 MUST be ignored.
65	4	File type data, normally expressed as four characters.

Field offset	Field length	Description
69	4	File creator data, normally expressed as four characters.
73	1	Finder flags, bits 15:8.
74	1	Pad, MUST be 0 (zero).
75	2	Icon vertical location, unsigned 16-bit integer.
77	2	Icon horizontal location, unsigned 16-bit integer.
79	2	File's folder ID.
812	1	File protected flag, low order bit.
82	1	Pad, MUST be 0.
83	4	Data fork length, signed 32-bit integer, zero if there is no data fork.
87	4	Resource fork length, signed 32-bit integer, zero if there is no resource fork.
91	4	File creation date, signed 32-bit integer representing a number of seconds since (or before, if negative) midnight, 01/01/2000, **UTC**.
95	4	File modification date, signed 32-bit integer representing a number of seconds since (or before, if negative) midnight, 01/01/2000, UTC.
99	2	Comment length, unsigned 16-bit integer, MUST be 0 (zero).
101	1	Finder flags, bits 7:0.
102	4	Signature. MIME reader MUST set this value to "mbin" (%x4d.62.69.6E).

Field offset	Field length	Description
106	1	File name script identifier. MIME reader SHOULD set to 0 (zero).
107		Extended finder flags, MIME reader SHOULD set to 0 (zero).
108	12	Zero fill.
120	2	Secondary header length, MUST be 0 (zero).
122	1	MacBinary version number. MUST be set to 130 (0x0082), indicating MacBinary III, when the MIME reader creates the **MacBinary** structure.
123	1	Minimum MacBinary version supported by this structure. MUST be set to 129 (0x0081), indicating MacBinary II, when the MIME reader creates the **MacBinary** structure..
124	2	CRC of previous 124 bytes. **MIME writers** SHOULD NOT validate this value. MIME readers SHOULD calculate this value by appling a Cyclic Redundancy Check algorithm on the first 124 bytes of the header. The CRC algorithm used by MacBinary is the CCITT algorithm, which uses the polynomial 0x1024. For more information on CRC-CCITT, see [X25].
Bytes 126:127		Zero fill.

When processing **AppleSingle** data, MIME readers MUST map **AppleSingle** fields to **MacBinary** fields as specified in the following table.

AppleSingleEntryId and type	MacBinary field	Comment

AppleSingleEntryId and type	MacBinary field	Comment
1, data fork	Bytes 83:86 – length; MacBinary data fork part	This mapping SHOULD only be used by MIME readers in MIME analysis of a stand-alone application/applefile, because (according to [RFC1740]) data fork SHOULD be in a separate MIME part in multipart/appledouble case.
2, resource fork	Bytes 87:90 – length; MacBinary resource fork part	If length is 0 (zero), MIME writers SHOULD NOT create this entry in AppleSingle.
3, ASCII string	Byte 1 – length, Bytes 2:64 – ASCII string value (only length bytes used)	File name. Note that MacBinary limits this string to 63 bytes. Excess bytes MUST be truncated.
8, ASFileDates structure, create	Bytes 91:94	File creation date, MIME readers SHOULD map it for AppleSingle to MacBinary conversion.
8, ASFileDates structure, modify	Bytes 91:94	File modification date, MIME readers SHOULD map it for AppleSingle to MacBinary conversion.
8, ASFileDates structure, access	None	MIME writers SHOULD set to 0 (zero) on conversion to AppleSingle.
8, ASFileDates structure, backup	None	MIME writers SHOULD set to 0 (zero) on conversion to AppleSingle.
9, ASFinderInfo structure, ioFlFndrInfo.fdType	Bytes 65:68	File type information.

AppleSingleEntryId and type	MacBinary field	Comment
9, ASFinderInfo structure, ioFlFndrInfo.fdCreator	Bytes 69:72	File creator information.
9, ASFinderInfo structure, ioFlFndrInfo.fdFlags	Byte 73 – bits 15:8 Byte 101 – bits 7:0	File finder flags word. MIME readers SHOULD map this element for AppleSingle to MacBinary conversion.
9, ASFinderInfo structure ioFlFndrInfo.fdLocation.v	Bytes 75:76	Icon vertical location MIME readers SHOULD map this element for AppleSingle to MacBinary conversion.
9, ASFinderInfo structure, ioFlFndrInfo.fdLocation.h	Bytes 77:78	Icon horizontal location. MIME readers SHOULD map this element for AppleSingle to MacBinary conversion.
9, ASFinderInfo structure, ioFlFndrInfo.fdFldr	Bytes79:80	File folder ID. MIME readers SHOULD map this element for AppleSingle to MacBinary conversion.
9, ASFinderInfo structure, ioFlXFndrInfo	None	MIME writers SHOULD fill with zeros on conversion to AppleSingle.
10, ASMacInfo structure, filler	None	MIME writers SHOULD fill with zeros on conversion to AppleSingle.
10, ASMacInfo structure, ioFlAttrib, bit 1	Byte 81, low order bit	Protected flag. MIME readers SHOULD map this element for AppleSingle to MacBinary conversion.

Conversion from a full **AppleSingle** structure, found in a stand-alone application/applefile MIME element that is not a child of multipart/appledouble, to a **reduced AppleSingle** structure that SHOULD be used as a child of multipart/appledouble, is done simply by removing the entry with **AppleSingleEntryId** equal to 1 (the data fork) and adjusting the **AppleSingle** header accordingly.

2.2.4.2.3 Application/Mac-binhex40

This section specifies **MIME analysis** for MIME parts with Content-Type application/mac-binhex40, as specified in [RFC1741].

The procedure of MIME header analysis for application/mac-binhex40 attachments is the same as for the procedure for ordinary file attachments that is specified in section 2.2.4.1, with the following exceptions:

1. **MIME reader**s MUST set the value of the **PidTagAttachMimeTag property** to "`application/mac-binhex40`".
2. The value of the Content-Transfer-Encoding **header field** MUST be ignored. MIME readers MUST use BinHex decoding, as specified in [RFC1741], instead.

Processing of the **MIME** body SHOULD include parsing a binary structure of the decoded content, as specified in [RFC1741]. MIME readers SHOULD use the **header** and resource fork data from this structure to fill the **PidNameAttachmentMacInfo** property with appropriate data, as specified in section 2.2.4.2.1. MIME readers SHOULD also use this data to fill the **MacBinary** structure, which SHOULD be written to the value of the **PidTagAttachDataBinary** property.

If parsing of the BinHex data fails, the entire message SHOULD be rejected by the MIME reader as not MIME compliant.

MIME readers SHOULD copy the attachment file name that is extracted from the **BinHex** structure to the value of **PidTagAttachFilename**, but only if no file name was found during analysis of the MIME headers. <104>

2.2.4.3 Attached Messages

If an attachment MIME part has its Content-Type **MIME header** set to message/rfc822 (or no Content-Type header is present, and this MIME part is a sub-part of the multipart/digest MIME part), **MIME reader**s SHOULD treat this attachment as an embedded message attachment, and set the value of the **Attachment object's PidTagAttachMethod property** to "5".

MIME analysis for MIME headers SHOULD be performed by the server in the same way as for ordinary file attachments, with the exception that the procedure for extracting the display name and file name for the attachment is different.

The display name for embedded message attachments is extracted from MIME part **header field**s in the following order:

1. If a Content-Type MIME header field is available on the attachment MIME part, and a non-empty *name* parameter is available on this header, its value SHOULD be used, else
2. If a Content-Disposition MIME header field is available on the attachment MIME part, and a non-empty *filename* parameter is available on this header, its value SHOULD be used, else
3. If a Content-Description MIME header field is available on the attachment, and its value is non-empty, it SHOULD be used, else
4. If a Subject header field is available on the attachments, and its value is non-empty, it SHOULD be used, else
5. The MIME reader SHOULD generate a name of its choosing.

The resulting value SHOULD be written to the **PidTagDisplayName** property on the attachment, and then processed further to obtain a valid file name, as follows:

1. All **Unicode** separator characters in the file name SHOULD be replaced with space characters.
2. All trailing and starting space and '.' characters SHOULD be removed.

The file name is then separated into base and extension parts. To do this, the server SHOULD look for the last occurrence of any of the following characters:

- backslash, "\\", U+005C
- forward slash, "/", U+002F
- colon, ":", U+003A
- period, ".", U+002E

If a "." (U+002E) character is the last one found, the part of the file name that precedes this character is considered to be base, and the rest is considered to be extension. In all other cases, extension is considered to be an empty string, and base part is considered to be the same as whole file name.

The resulting file name value SHOULD be written to the **PidTagAttachLongFilename** property, and the resulting extension value SHOULD be saved in the **PidTagAttachExtension** property. <105> The file name SHOULD then be processed further to obtain a valid 8.3 file name, as follows:

1. The value SHOULD be first separated into base and extension parts, by using the last "." character as a separator (if no such character is present, the extension is considered to be empty; the separator character itself is not included in the name or extension).

2. "+", ",", "=", "[", "]", and ";" characters SHOULD be replaced with the "_" (underscore) character.

3. Space, ".", "\","\", "*", "<", ">", "?", ":", and "|" characters, as well as characters with UTF8 code greater than 127, SHOULD be removed.

6. If the base becomes an empty string, some default, non-empty value SHOULD be used.

4. The base part of the file name SHOULD be trimmed to 8 characters; the extension part – to 3 characters.

5. If either the name or the extension changed, the base part SHOULD additionally be trimmed to 6 characters, and "~1" SHOULD be added to its end.

6. The file name is saved in the <base>.<extension> format.

The resulting file name SHOULD be written to **PidTagAttachFilename**.

The MIME part body for this attachment SHOULD be used for further MIME analysis that SHOULD result in assigning values to the properties of the embedded message from this attachment. This MIME analysis is performed in a way that is similar to that for ordinary MIME messages, with the following exceptions:

1. The X-MS-Exchange-Organization-Original-Sender MIME header value SHOULD be saved in the **PidNameQuarantineOriginalSender** property, if the header value is present. <106><107>

2. Unknown MIME headers, starting with "X-MS-Exchange-Organization-" or "X-MS-Exchange-Forest-", SHOULD NOT be excluded from analysis. <108>

3. After MIME analysis is done for the embedded message, the **PidTagMessageFlags** property SHOULD be modified; the IsDraft flag ("0x8") SHOULD be removed, and the IsRead flag (0x1) SHOULD be set.

2.2.5 External Body Attachments

Attachment **MIME** parts with a Content-Type MIME **header field** set to "`message/external-body`" SHOULD be analyzed in the same way as ordinary file attachments, with the exceptions specified in this section. <109>

If the Content-Type MIME header field has no *access-type* parameter, or if the value of that parameter is not "`anon-ftp`", **MIME reader**s SHOULD save the entire MIME part in the **PidTagAttachDataBinary property** on the attachment.

Otherwise, the following differences in **MIME analysis** apply:

- Different file name extraction logic SHOULD be applied, as specified later in this section.
- The server SHOULD ignore the MIME part body. Instead, a specially formatted URL data string is saved in the **PidTagAttachDataBinary** property in **ASCII** format, as specified in this section.
- In this case, MIME readers expect the *name*, *site*, *directory*, and *mode* parameters to be present in the Content-Type MIME header. Clients SHOULD NOT create MIME that does not meet this criteria.

The URL data string to save in the **PidTagAttachDataBinary** property is constructed as follows:

```
"[Internet Shortcut]" CR LF "URL=ftp://" site "/" directory "/" name [mode]

;contains header field "site" parameter value

site=1*xchar ; xchar defined in [RFC1738]


;contains header field "directory" parameter value

directory=1*xchar

;contains header field "name" parameter value

name=1*xchar

;if header field "mode" parameter is "ascii", contains ";type=a"

;if header field "mode" parameter is "image, contains ";type=i"

;otherwise not present

mode=";type=" 1*lowalpha ; lowalpha defined in [RFC1738]
```

The file name extraction logic is similar to that for ordinary file attachments, with the following exceptions:

1. MIME readers MUST use the *name* parameter value from the Content-Type MIME header as a value of the attachment file name. The file extension (a part of the file name after the last appearance of the '.' character) SHOULD be replaced with "url"; if the original file name has no extension, ".url" SHOULD be added at the end of the file name string.
2. The sanitizing logic specified in section 2.2.4.1.1 SHOULD NOT be applied in this case.

3. The file name value constructed in Step 1 SHOULD be used as an attachment display name as well.
4. All additional filtering logic specified in section 2.2.4.1.1 still applies in this case.
5. The procedure for calculating a value for the **PidTagAttachFilename** property is the same as for embedded message attachments.

2.3 Additional Content Types

2.3.1 Analysis of Non-MIME Content

Internet message content that lacks a MIME-Version **header field** MAY still be supported by **MIME reader**s. The absence of a MIME-Version header field makes the payload of **SMTP** or elsewhere non-**MIME**, with different behavior for inline attachments; header fields such as Content-Type, Content-Disposition, and such have no special meaning. MIME readers MAY, nevertheless, assume the presence of a MIME-Version field and treat header fields such as Content-Type in the usual way.

The following is an example of such a message.

```
From: <user1@example.com>
To: <user2@example.com>
Subject: Example Legacy 822 message with attachment.
Date: Mon, 10 Mar 2008 14:36:46 -0700

this is a test message

begin 664 Flag.png
MB5!.1PT*&@H````-24A$4@```!0````-"`8```"I4$Y>````!F)+1T0`_P#_
M`/^@O:>3````"7!(67,```L3```+$P$`FIP8````!W1)344'V`,+`!8G&XK)
MCP```1M)1$%4.,NMDTUJPE`4A;\7\Z,ET8$-07`@BE,1.Q)T#2*X!-V+6^@J
M7(2.Q14X<J+6@29$DZBOHTJA)GW0GO&Y'^=>SA6\O4O^49J*R;4#7#OX.W#2
MGU,I'6EZ6YK>EDKIR*0_SP2*M)6=_(6"D1!$%F%L`O!BQMA6Q#DQ\"]Y]82M
MZH919\G.=QXP@#`VV?D.H\Z25G6CGM#(W4ANN<S5TCP_$FI",AW.L/1K*LS2
MKTR',S0AU1+FM#NW>W8!TCQ/IQKNGG%OD0H;]Q8TW+WZ#0M&@A`2VXKX"&SN
M4CS.\6H'!)&%E()S8J@E/"<&86PR:*^HE0]TZVNZ]36U\H%!>T48FT]AF3W\
F+J]X`F![*O[Z*;K*.ZF`OO0)9G1H4-$`H@T`````245.1*Y"8((`
`
end

begin 664 Flag.png
MB5!.1PT*&@H````-24A$4@```!0````-"`8```"I4$Y>````!F)+1T0`_P#_
M`/^@O:>3````"7!(67,```L3```+$P$`FIP8````!W1)344'V`,+`!8G&XK)
MCP```1M)1$%4.,NMDTUJPE`4A;\7\Z,ET8$-07`@BE,1.Q)T#2*X!-V+6^@J
M7(2.Q14X<J+6@29$DZBOHTJA)GW0GO&Y'^=>SA6\O4O^49J*R;4#7#OX.W#2
```

```
MGU,I'6EZ6YK>EDKIR*0_SP2*M)6=_(6"D1!$%F%L`O!BQMA6Q#DQ\"]Y]82M
MZH919\G.=QXP@#`VV?D.H\Z25G6CGM#(W4ANN<S5TCP_$FI",AW.L/1K*LS2
MKTR',S0AU1+FM#NW>W8!TCQ/IQKNGG%OD0H;]Q8TW+WZ#0M&@A`2VXKX"&SN
M4CS.\6H'!)&%E()S8J@E/"<&86PR:*^HE0]TZVNZ]36U\H%!>T48FT]AF3W\
F+J]X`F![*O[Z*;K*.ZF`OO0)9G1H4-$`H@T`````245.1*Y"8((`
`
end
```

MIME writers SHOULD NOT generate messages in this format; MIME SHOULD be generated instead. MIME readers SHOULD analyze messages in this format into header fields, **plain text** body, and attached files (possibly including a **TNEF** attachment).

2.3.2 Message/Partial

The message/partial content type is not supported. **MIME reader**s MUST reject messages that contain **MIME** entities with a message/partial Content-Type **header field**. This is to prevent virus scanning from being defeated by splitting up attachment content.

2.3.3 Multipart/Digest

This content type is treated exactly as multipart/mixed, except that the assumed Content-Type for **body parts** with no Content-Type **header field** SHOULD be message/rfc822 rather than text/plain.

3 Structure Examples

This example shows a very simple e-mail message in both **MIME** and **Message object** formats. The following is the message in MIME format:

```
Received: from mailer01.example.com by mailer02.example.com
  with Microsoft SMTP Server; Mon, 11 Feb 2008 14:45:44 -0800
From: <user1@example.com>
To: <user2@example.com>; <user3@example.com>
Subject: test message
Date: Mon, 11 Feb 2008 14:45:32 -0800
Message-ID: <000001c86cff$cf0dd670$ae62379d@mail.example.com>
MIME-Version: 1.0
Content-Type: text/plain
Content-Transfer-Encoding: 7bit
Importance: normal
Priority: normal

this is a test message
```

The following table shows this simple message represented as a Message object. The message itself has several **properties**, and it contains **recipients** each with several properties of its own. The recipients are shown in the table that follows.

Message Property	Type	Value
PidTagMessageDeliveryTime	**PtypTime**	%xF9.2D.82.D6.FF.6C.C8.01
PidTagSentRepresentingName	**PtypString**	Test user 1
PidTagSentRepresentingAddressType	**PtypString**	SMTP
PidTagSentRepresentingEmailAddress	**PtypString**	user1@example.com
PidTagSubject	**PtypString**	test message
PidTagClientSubmitTime	**PtypTime**	%x00.8E.AD.CE.FF.6C.C8.01
PidTagInternetMessageId	**PtypString**	<000001c86cff$cf0dd670$ae62379d@mail.example.com>
PidTagImportance	**PtypInteger32**	1
PidTagPriority	**PtypInteger32**	0
PidTagBody	**PtypString**	this is a test message
PidTagInternetCodepage	**PtypInteger32**	28591

Message Property	Type	Value
PidTagObjectType	**PtypInteger32**	5
PidTagMessageFlags	**PtypInteger32**	0x23

Row ID	Recipient Property	Type	Value
38714304	**PidTagDisplayName**	**PtypString**	Test user 2
38714304	**PidTagAddressType**	**PtypString**	EX
38714304	**PidTagEmailAddress**	**PtypString**	/O=Example1/OU= Administrative Group/cn=Recipients/cn=user2
38714304	**PidTagSmtpAddress**	**PtypString**	user2@example.com
38714304	**PidTagRecipientType**	**PtypInteger32**	1
38714304	**PidTagObjectType**	**PtypInteger32**	6
38714305	**PidTagDisplayName**	**PtypString**	Test user 3
38714305	**PidTagAddressType**	**PtypString**	EX
38714305	**PidTagEmailAddress**	**PtypString**	/O=Example1/OU= Administrative Group/cn=Recipients/cn=user3
38714305	**PidTagSmtpAddress**	**PtypString**	user3@example.com

Row ID	Recipient Property	Type	Value
38714305	**PidTagRecipientType**	**PtypInteger32**	1
38714305	**PidTagObjectType**	**PtypInteger32**	6

While obviously less compact than the **MIME** format, the Message object format makes strongly typed data available. Both client and server code can sort, find, and process messages according to specific criteria such as "all unread messages," "all messages tagged as Personal," or "all calendar items occurring in the week of 2/12/2008, sorted by start time."

4 Security Considerations

4.1 Unsolicited Commercial E-mail (Spam)

A significant business has evolved around the sending of unsolicited commercial e-mail (colloquially referred as **spam**). Unlike physical bulk mail where there exists built-in restrictions on who can send and what they have to pay, the general structure of **SMTP** allows anonymous sources to send e-mail virtually without restriction. Attempts are being made to reduce the volume of spam that makes it to a person's mailbox, but care has to be taken to not affect legitimate senders.

Part of the success of this industry is the fact that people impute importance to unverifiable things (see [MS-OXCSPAM]). For example, the purported sender of a piece of mail (considering most mail is not digitally signed) is commonly used by people to attach importance and priority. If the message appears to come from a person's boss, there is a higher probability that the employee would act on the message. In this case, care SHOULD be taken when receiving mail over unauthenticated transports that, although the routing address of the sender matches a valid employee or contact, the address stored on the **Message object** remain as the external routing address, and not be replaced with its **address book** equivalent, which would convey more importance to the content.

4.2 Information Disclosure

Content that is sent can contain hints about the source network's topology and structure. In **MIME**, this can be discerned from the Received **headers** (every **SMTP** server and potentially the client's computer are listed by its network address). In addition, if the optional algorithm specified in [RFC2822 section 3.6.4] to generate a unique Message-Id **header** value is implemented, the client's network address and internal domain name is exposed. Alternately, a **GUID** can be used to satisfy the unique identifier, circumventing

the first data exposure. The aforementioned problem also exists for Content-Id header and *boundary* parameter values. It is suggested that a GUID be used here as well.

Additionally, when sending **recipient** data other than the properties mentioned previously, implementations SHOULD be aware that internal data can be exposed. For example, office numbers and phone numbers could be cached on each recipient. This is an issue on embedded messages that are transported via **TNEF** (see [MS-OXTNEF]), as TNEF has the ability to carry more recipient information than is available with MIME headers.

Care SHOULD also be taken when receiving mail to deal with some information disclosure issues. If the e-mail message leverages any feature that requires the client to "fetch" additional resources when displaying it, the act of fetching can expose the fact that the recipient is an actual employee, and the date and time that the message was read. Examples of this are external bodies, **HTML** stylesheets, and images.

4.3 Content-Type Versus File Extension Mismatch

Various clients accept the Content-Type received from the server, but then verify the content. This can include checking the file extension or looking for a thumbprint at the start of the file, and then mapping this data back to a verified Content-Type. If the file extension or thumbprint does not match the stated Content-Type, a Content-Type value derived from the file extension or thumbprint is used instead. This behavior is actually codified in [RFC1521], which allows the sender to set the Content-Type to "`application/octet-stream`" (or not set it at all). The recipient is then responsible for correctly determining the type of content via alternate means.

In addition, it was found that various clients incorrectly set the Content-Type either by mistake or intentionally. Support to address the former has existed for quite some time but has opened a path to potentially thwart policy scanning and protection applications running on the server.

Therefore **MIME reader**s MAY want to correct mislabeled Content-Type **header** values so that server Policy Agents and clients can trust the header value. Clients SHOULD offer a mechanism to do one or more of the following:

- Suppress correcting the Content-Type **header**.
- Block attachments by type or extension.
- Offer some sort of security barrier before running any script, or binary.

These steps are particularly important if the sender is unauthenticated.

4.4 Do Not Support Message/Partial

A Content-Type of "message/partial" allows large messages to be sent in pieces and re-assembled by the client. It was originally designed to work around transmission failures during slow delivery causing the complete message to be resent from scratch, and to work around message size restrictions of implementations of protocols like **SMTP**. With increased bandwidth speeds, and greater connectivity, the long transmission times are more a thing of the past. Continued support for this Content-Type allows an avenue for content that is inappropriate to reach (or leave) the e-mail client's computer. This could include things such as "Information disclosure" of proprietary information, unsolicited commercial e-mail (**spam**), and computer virus attachments.

E-mail servers attempt to protect their users from inappropriate content by implementing Policy applications that run as part of the protocol. For them to work efficiently, the complete content MUST be incorporated into one message. For this reason, servers SHOULD prohibit sending or receiving messages with a Content-Type of "message/partial".

4.5 Considerations for Message/External-Body

The original **MIME** RFC [RFC1521] allowed the body of an entity to be referenced externally rather than requiring it to be inline. The current MIME RFC [RFC2046 section 5.2.3] specifies the form of this construct; the security implications are as follows:

1. The blind retrieval of the content by the client can disclose information about the **recipient**.
2. The authentication mechanism tied to the retrieval (*access-type* parameter) can result in a pop-up dialog box, leading the user to expose credential information.
3. The server (Policy or delivery application) that is attempting to check the content opens up a denial of service vector for the remote host to tie up server resources.

4.6 Preventing Denial of Service Attacks

4.6.1 Submission Limits

Servers MAY limit the size of received messages to limit resource consumption. Such limits can be different for authenticated versus anonymous senders.

4.6.2 Complexity of Nested Entities

It is possible to represent very complex hierarchies with **MIME** and to add superfluous entity layers (Content-Type: multipart/). Servers and clients SHOULD protect themselves from stack overflows or heap starvation. This MAY involve limiting the nesting depth of attachments and **body parts** within a single message.

4.6.3 Number of Embedded Messages

A server that implements this protocol converts **MIME** content into a **Message object** representation. This causes each embedded message to be mapped individually and all attachments to be included therein. One implementation of this is to recursively handle each attached embedded message, but care SHOULD be taken not to encounter a stack overflow by doing so.

4.6.4 Compressed Attachments

Analyzing each attachment on the server is a concern when decompression is required. It is possible to encounter compressed content that requires large volumes of disk space, memory, or other resources, leading to a denial of service.

5 Appendix A: Office/Exchange Behavior

The information in this specification is applicable to the following versions of Office/Exchange:

- Microsoft Office Outlook 2003
- Microsoft Exchange Server 2003
- Microsoft Office Outlook 2007
- Microsoft Exchange Server 2007

Exceptions, if any, are noted as follows. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Office/Exchange behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies Office/Exchange does not follow the prescription.

<1> Section 2.1: Outlook 2003 SP3 and Outlook 2007 SP1 do not use the HTML only setting.

<2> Section 2.1.1: Outlook 2003 SP3 and Outlook 2007 SP1 do not encapsulate addresses.

<3> Section 2.1.1.1: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate **MIME recipients** for **Bcc recipients** if it is generating a message to send via **SMTP**.

<4> Section 2.1.1.1: Outlook 2003 SP3 and Outlook 2007 SP1 use the listed steps in the following order: 4, 5, 1, 2, 3, 7. Outlook 2003 SP3 and Outlook 2007 SP1 do not use step 6.

<5> Section 2.1.1.1.1: Exchange 2003 SP2 generates **attRecipTable** for top-level summary and legacy **TNEF** messages.

<6> Section 2.1.1.2: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the values of **PidTagReplyRecipientEntries** and **PidTagReplyRecipientNames** to the **TNEF body part**.

<7> Section 2.1.1.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the values of the **PidTagSentRepresenting** property group to the **TNEF body part**.

<8> Section 2.1.1.4: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the values of the **PidTagSenderName** property group to the **TNEF body part**.

<9> Section 2.1.1.5: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the values of **PidTagOriginatorDeliveryReportRequested** and the **PidTagSentRepresenting** property group to the **TNEF body part**.

<10> Section 2.1.1.6: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the values of the **PidTagReadReceipt** and **PidTagSentRepresenting** groups of properties. Note that they do copy **PidTagReadReceiptRequested**.

<11> Section 2.1.1.8: Outlook 2003 SP3 and Outlook 2007 SP1 do not encapsulate addresses.

<12> Section 2.1.1.8: Exchange 2007 SP1 encodes only the address part and uses the address-type as-is. On decoding, Exchange 2007 SP1 scans for the first hyphen ("-") after "IMCE" and uses the prefix as-is without parsing for any escaped characters. Exchange 2007 SP1 has no limitation on the length of address-type. Exchange 2007 SP1 does not properly de-encapsulate if address-type contains an ASCII hyphen ("-").

<13> Section 2.1.1.8: Exchange 2003 SP2 builds the entire string including address-type and address, and then encodes the whole string, escaping any non-alphanumeric characters contained in **address-type**. On decoding, Exchange 2003 SP2 unescapes the entire string and scans the first nine characters for a hyphen ("-"). If a hyphen is not found, then the address is not de-encapsulated. The **address-type** is limited to eight characters. Only ASCII alphanumeric characters are allowed in **address-type**. Exchange 2003 SP2 does not properly de-encapsulate if **address-type** contains an ASCII hyphen ("-").

<14> Section 2.1.2: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy properties to the **TNEF body part** if there is a corresponding **MIME header field**.

<15> Section 2.1.2.1: Outlook 2003 SP3 and Outlook 2007 SP1 do not convert appointment items to **MIME**.

<16> Section 2.1.2.2: Exchange 2003 SP2 does not generate the Content-Class **header field**.

<17> Section 2.1.2.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-CallingTelephoneNumber **header**.

<18> Section 2.1.2.3: Exchange 2003 SP2 does not generate the X-CallingTelephoneNumber **header**.

<19> Section 2.1.2.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-VoiceMessageDuration **header**.

<20> Section 2.1.2.3: Exchange 2003 SP2 does not generate the X-VoiceMessageDuration **header**.

<21> Section 2.1.2.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-VoiceMessageSenderName **header**.

<22> Section 2.1.2.3: Exchange 2003 SP2 does not generate the X-VoiceMessageSenderName **header**.

<23> Section 2.1.2.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-FaxNumberOfPages **header**.

<24> Section 2.1.2.3: Exchange 2003 SP2 does not generate the X-FaxNumberOfPages **header**.

<25> Section 2.1.2.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-AttachmentOrder **header**.

<26> Section 2.1.2.3: Exchange 2003 SP2 does not generate the X-AttachmentOrder **header**.

<27> Section 2.1.2.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-CallID **header**.

<28> Section 2.1.2.3: Exchange 2003 SP2 does not generate the X-CallID **header**.

<29> Section 2.1.2.7: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy **PidTagClientSubmitTime** to the **TNEF body part**.

<30> Section 2.1.2.8: Outlook 2003 SP3 and Outlook 2007 SP1 instead copy **PidTagSubject** to the Subject **header field**.

<31> Section 2.1.2.8: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the message subject to the **TNEF body part**.

<32> Section 2.1.2.11: Outlook 2003 SP3 and Outlook 2007 SP1 will generate a new Message-ID if it is generating a message to send via **SMTP**.

<33> Section 2.1.2.14: Exchange 2003 SP2 does not copy the value of the PidTagInReplyToId property to the In-Reply-To **header**.

<34> Section 2.1.2.16: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the Accept-Language **header**.

<35> Section 2.1.2.16: Exchange 2003 SP2 does not generate the Accept-Language **header**.

<36> Section 2.1.2.16: Exchange 2003 SP2 does not generate the Content-Language **header** field.

<37> Section 2.1.2.17: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate any classification **headers**.

<38> Section 2.1.2.17: Exchange 2003 SP2 does not generate any classification **headers**.

<39> Section 2.1.2.18: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-Payload-Provider-Guid **header**.

<40> Section 2.1.2.18: Exchange 2003 SP2 does not copy the value of the **PidTagAttachPayloadProviderGuidString** property to the X-Payload-Provider-Guid**header**.

<41> Section 2.1.2.18: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-Payload-Class **header**.

<42> Section 2.1.2.18: Exchange 2003 SP2 does not copy the value of the **PidTagAttachPayloadClass** property to the X-Payload-Class header.

<43> Section 2.1.2.19: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-MS-HasAttach **header**.

<44> Section 2.1.2.20: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-Auto-Response-Suppress **header**.

<45> Section 2.1.2.20: Exchange 2003 SP2 does not generate the X-Auto-Response-Suppress **header**.

<46> Section 2.1.2.21: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-MS-Exchange-Organization-AutoForwarded **header**.

<47> Section 2.1.2.21: Exchange 2003 SP2 does not generate the X-MS-Exchange-Organization-AutoForwarded **header**.

<48> Section 2.1.2.22: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-MS-Exchange-Organization-SenderIdResult **header**.

<49> Section 2.1.2.22: Exchange 2003 SP2 does not generate the X-MS-Exchange-Organization-SenderIdResult **header**.

<50> Section 2.1.2.23: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-MS-Exchange-Organization-PRD **header**.

<51> Section 2.1.2.23: Exchange 2003 SP2 does not generate the X-MS-Exchange-Organization-PRD **header**.

<52> Section 2.1.2.24: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the X-MS-Exchange-Organization-SCL **header**.

<53> Section 2.1.2.24: Exchange 2003 SP2 does not generate the X-MS-Exchange-Organization-SCL **header**.

<54> Section 2.1.2.28: Outlook 2003 SP3 and Outlook 2007 SP1 also set **PidTagTransportMessageHeaders** when downloading messages via **POP3** or **IMAP**.

<55> Section 2.1.3.1: Outlook 2003 SP3 and Outlook 2007 SP1 either convert the **RTF** text to **HTML** or generate a **TNEF** attachment that contains the RTF body.

<56> Section 2.1.3.2: This is for compatibility with the Exchange 2003 SP2 **TNEF** reader, which loses body text if only a **plain text** body is encoded in TNEF.

<57> Section 2.1.3.4: Exchange 2003 SP2 relies on store for text conversions, so it does the MAY behavior rather than the SHOULD behavior.

<58> Section 2.1.3.6: Exchange 2007 SP1 does not check to determine whether an apparently inline attachment is, in fact, referenced from the message body.

<59> Section 2.1.3.8.2: Exchange 2003 SP2 relies on the store for text conversions, so it performs the MAY behavior rather than the SHOULD behavior.

<60> Section 2.1.4.1: Exchange 2007 SP1 does not exclude attached **Message objects** or attachments to **plain text** messages from being considered inline.

<61> Section 2.1.4.1.1: Exchange 2007 SP1 does not exclude non-OLE attachments in an **RTF** message from being considered inline.

<62> Section 2.1.4.1.2: The **MIME writer** in Exchange 2007 SP1 only checks condition 1, not conditions 2 or 3, when classifying attachments as inline.

<63> Section 2.1.4.2.2: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the Content-Description **header**.

<64> Section 2.1.4.2.2: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the *size* parameter for the Content-Disposition **header**.

<65> Section 2.1.4.2.2: Exchange 2003 SP2 does not generate the *size* parameter for the Content-Disposition **header**.

<66> Section 2.1.4.2.2: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the *creation-date* parameter for the Content-Disposition **header**.

<67> Section 2.1.4.2.2: Exchange 2003 SP2 does not generate the *creation-date* parameter for the Content-Disposition **header**.

<68> Section 2.1.4.2.2: Exchange 2007 SP1 appends the word "GMT" to the *creation-date* parameter without adjusting the time from the server timezone to GMT.

<69> Section 2.1.4.2.2: Outlook 2003 SP3 and Outlook 2007 SP1 do not generate the *modification-date* parameter for the Content-Disposition **header**.

<70> Section 2.1.4.2.2: Exchange 2003 SP2 does not generate the *modification-date* parameter for the Content-Disposition **header**.

<71> Section 2.1.4.2.2: Exchange 2007 SP1 appends the word "GMT" to the *modification-date* parameter without adjusting the time from the server timezone to GMT.

<72> Section 2.1.4.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not encode attachments in the MacBinary format.

<73> Section 2.1.4.3: Exchange 2007 SP1 does not ignore the presence of secondary header data in a MacBinary stream; it fails if secondary header data is present.

<74> Section 2.1.4.3: Exchange 2007 SP1 does not ignore the presence of additional data in a MacBinary stream; it fails if additional data is present.

<75> Section 2.1.4.4: Outlook 2003 SP3 and Outlook 2007 SP1 do not convert OLE attachments. OLE attachments are omitted from the **MIME** version of the message.

<76> Section 2.2.1.1: Outlook 2003 SP3 and Outlook 2007 SP1 do not check for IMCEA encapsulation and do not perform de-encapsulation.

<77> Section 2.2.1.2: Outlook 2003 SP3 and Outlook 2007 SP1 do not check for IMCEA encapsulation and do not perform de-encapsulation.

<78> Section 2.2.1.3: Outlook 2003 SP3 and Outlook 2007 SP1 will not use the **attSentFor** attribute or the **PidTagSentRepresenting** value if the From **header** is absent.

<79> Section 2.2.2: Exchange 2007 SP1 uses the last instance to set the value of the corresponding property.

<80> Section 2.2.2.11: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the value of the Accept-Language or X-Accept-Language **header**.

<81> Section 2.2.2.11: Exchange 2003 SP2 does not copy the value of the Accept-Language or X-Accept-Language **header**.

<82> Section 2.2.2.13: Exchange 2007 SP1 uses whichever **header** shows up last in the list of header information, either the Expires header or the Expiry-Date header, as the header used to set **PidTagExpiryTime**.

<83> Section 2.2.2.14: Outlook 2003 SP3 and Outlook 2007 SP1 ignore the X-AUTO-Response-Suppress and Precedence **headers**.

<84> Section 2.2.2.14: Exchange 2003 SP2 ignores the X-AUTO-Response-Suppress and Precedence **headers**.

<85> Section 2.2.2.15: Exchange 2003 SP2 does not generate the Content-Class **header field**.

<86> Section 2.2.2.15: Outlook 2003 SP3 and Outlook 2007 SP1 do no special processing for "urn:content-class:custom."

<87> Section 2.2.2.15: Exchange 2003 SP2 does no special processing for "urn:content-class:custom."

<88> Section 2.2.2.18: Exchange 2003 SP2 does not write an X-Payload-Class and an X-Payload-Provider-Guid **header field**.

<89> Section 2.2.2.19: Outlook 2003 SP3 and Outlook 2007 SP1 do not read or set any of the classification **headers**.

<90> Section 2.2.2.19: Exchange 2003 SP2 does not read or set any of the classification **headers**.

<91> Section 2.2.2.20: Outlook 2003 SP3 and Outlook 2007 SP1 do not read or set any of the unified messaging **headers**.

<92> Section 2.2.2.20: Exchange 2003 SP2 does not read or set any of the unified messaging **headers**.

<93> Section 2.2.2.21: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the value of the Content-ID header to **PidTagBodyContentId** and do not copy the value of **PidTagBodyContentID** to the Content-ID **header**.

<94> Section 2.2.2.21: Exchange 2003 SP2 does not copy the value of the Content-ID **header** to **PidTagBodyContentId** and does not copy the value of **PidTagBodyContentID** to the Content-ID header.

<95> Section 2.2.2.22: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the value of a Content-Base **header** to the **PidNameContentBase** property.

<96> Section 2.2.2.23: Outlook 2003 SP3 and Outlook 2007 SP1 do not copy the value of a Content-Location **header** to the **PidTagBodyContentLocation** property.

<97> Section 2.2.2.26: Both Exchange 2003 SP2 and Exchange 2007 SP1 cannot create the named property corresponding to a generic header field if (1) the mailbox database named property quota has been exceeded; or (2) measures adopted to prevent exhausting the named property quota interfere with creating the named property.

<98> Section 2.2.2.26: Outlook 2003 SP3 and Outlook 2007 SP1 do not create named properties for **headers**.

<99> Section 2.2.4.1.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not use the parameters of the Content-Disposition **header** to set creation or modification dates.

<100> Section 2.2.4.1.4: The **MIME reader** in Exchange 2007 SP1 does not verify that inline attachment candidates are in fact referenced from the message body before marking them as inline.

<101> Section 2.2.4.2.1: Exchange 2007 SP1 gets the attachment file name from **AppleSingle** data, instead of preferring a file name found in **MIME headers**.

<102> Section 2.2.4.2.2: Exchange 2007 SP1 gets the attachment file name from **AppleSingle** or MacBinary data, instead of preferring a file name found in **MIME headers**.

<103> Section2.2.4.2.2: Exchange 2007 SP1 does not support a **MacBinary** structure with additional **header** data or comment part present.

<104> Section 2.2.4.2.3: Exchange 2007 SP1 gets the attachment file name from BinHex data, instead of preferring a file name found in **MIME headers**.

<105> Section 2.2.4.3: Exchange 2003 SP2 does not save the resulting extension value in the **PidTagAttachExtension** property.

<106> Section 2.2.4.3: Outlook 2003 SP3 and Outlook 2007 SP1 do not save the X-MS-Exchange-Organization-Original-Sender **header** value.

<107> Section 2.2.4.3: Exchange 2003 SP2 does not save the X-MS-Exchange-Organization-Original-Sender **header** value.

<108> Section 2.2.4.3: Outlook 2003 SP3 and Outlook 2007 SP1 exclude unknown **MIME headers** that start with "X-MS-Exchange-Organization-" or "X-MS-Exchange-Forest-" from analysis.

<109> Section 2.2.5: Outlook 2003 SP3 and Outlook 2007 SP1 analyze attachment **MIME** parts with the Content-Type set to "message/external-body" the same as they do ordinary file attachments. No special analysis is performed.

Index