[MS-OXTNEF]: Transport Neutral Encapsulation Format (TNEF) Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights**. This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.

- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.

- **Patents**. Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp/default.mspx). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@microsoft.com.

- **Trademarks**. The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

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

Preliminary Documentation. This documentation is preliminary documentation for these protocols. Since the documentation may change between this preliminary version and the final version, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Tools. This protocol documentation is intended for use in conjunction with publicly available standard specifications and networking programming art, and assumes that the reader is either familiar with the aforementioned material or has immediate access to it. A protocol specification does not require the use of Microsoft programming tools or programming environments in order for a Licensee to develop an implementation. Licensees who have access to Microsoft programming tools and environments are free to take advantage of them.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability

Table of Contents

1 Introduction

This document specifies an encoding method for transmitting rich properties of electronic mail messages over a serial data stream. The result might be transported as a stream, as a file attachment in an arbitrary transport, or as a MIME [RFC2045] entity body on an internet transport.

1.1 Glossary

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

binary large object (BLOB)
code page
little-endian
message object
metafile
named property
property
Transport Neutral Encapsulation Format (TNEF)

The following terms are specific to this document:

CLSID: A GUID associated with a COM class.

Interface Identifier (IID): A GUID that uniquely indentifies a particular COM interface.

TNEF Reader: The entity that is decoding a TNEF structure after message reception, for the purpose of reconstructing the rich properties that are contained in the stream.

TNEF Writer: The entity that is encoding/building a TNEF structure for the purpose of transporting rich properties.

UTF-16LE (Unicode Transformation Format, 16-bits, Little-Endian): The encoding scheme specified in [UTF16] for encoding Unicode (see [Unicode30]) characters as a sequence of 16-bit codes, each encoded as two 8-bit bytes with least-significant-byte first.

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-OAUT] Microsoft Corporation, "OLE Automation Protocol Specification", March 2007, http://go.microsoft.com/fwlink/?LinkId=112419.

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

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

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

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

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

[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.

[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.

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

[UTR#17] Whistler, K., Davis, M., Freytag, A., "Unicode Technical Report #17: Character Encoding Model", September 2004, http://www.unicode.org/reports/tr17.

1.2.2 Informative References

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

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

1.3 Structure Overview (Synopsis)

Transport Neutral Encapsulation Format (TNEF) organizes a hierarchy of rich message properties into a flattened structure which can be represented as a serial data stream. The typical structure of a particular property within the stream is: identifier (which usually also includes type information), size (where not exactly determined by type), data. In some cases, as documented below, groups of properties or multiple-value properties include counts. Others might include "padding" to enforce a particular alignment of the data.

1.4 Relationship to Protocols and Other Structures

The structure is intended to permit the transmission of rich message property information over transports that have no mechanism for representing that information natively.

The structure can be included as a file attachment (winmail.dat), as a MIME [RFC2045] body part (using the "application/ms-tnef" media type), added to the transmitted plaintext body using UUENCODE or a similar method, to be decoded at the recipient end, or transported from sender to recipient using whatever means are provided by the protocol employed in transmitting message information between them.

This document specifically covers the application of this structure for transmission of message data over SMTP, POP, IMAP or other internet protocols that incorporate MIME [RFC2045].

All numeric datatypes in the structure greater than one byte in size are **little-endian** and any handling of these values on platforms that are not **little-endian** are required to take this into account and perform the appropriate transformations to get correct numbers, counts, values, etc.

Contents of example strings in this document will be shown in ABNF [RFC4234] format. Where the string has a terminating zero character, this will also be in that format. An example would be "user1@example.com" %x00. For the purpose of example strings, the different terminating zero character size in a UNICODE character set will not be illustrated.

1.5 Applicability Statement

The original application of the structure was in the Microsoft Mail 3.0 Windows client, to permit the creation and representation of "Message Classes" [MS-OXGLOS] other than simple e-mail messages, and some additional attributes that were not natively supported by the transport protocol.

This application was further extended to allow the transport of the rich set of properties required by applications such as Microsoft Outlook, including named properties [MS-OXPROPS]. For backwards compatibility with the original implementation, a special attribute is used to encapsulate the new message properties, and those properties with analogues to the original implementation are usually represented using the original attribute syntax.

To avoid confusion, attributes using the original syntax will be referred to as "attributes" and the rich set of properties will be referred to as "properties" for the remainder of this document.

1.6 Versioning

None.

2 Structure

The TNEF stream starts with a signature and a legacy key value, an attribute containing a legacy version number, and an attribute containing the Windows codepage used by the encoder for ANSI attributes and properties. After that, the stream is a series of attributes laid out, one after the other – message attributes followed by attachment attributes. There are 3 special attributes that contain the various message and attachment properties, two of which are counted lists; the third is a counted list of counted lists.

Each attribute is laid out as follows: id, length (of the contained data), the data itself, and a simple 16-bit checksum of the bytes comprising the data.

Encapsulated properties SHOULD be encoded after the attributes. Values of encapsulated properties SHOULD be used instead of any conflicting mapped attribute values. Message attributes and properties SHOULD be encoded before attachment attributes and properties.

Each set of attachment attributes MUST begin with attAttachRendData, followed by any other attributes; attachment properties encoded in attAttachment SHOULD be last.

2.1 Additional Terms Used in this Document

2.1.1 Address Representations

Address elements other than recipients, such as From and Sender, are represented in a message object by a group of four properties: display name, address type, e-mail address, entry ID. In subsequent sections these groups are referred to as described here.

PidTagRcvdRepresenting_XXX: Refers to **(PidTagRcvdRepresentingName, PidTagRcvdRepresentingEmailType and PidTagRcvdRepresentingEmailAddress),** or **PidTagRcvdRepresentingEntryId**, where either would suffice to fully represent the display name, e-mail transport type, and e-mail address of a particular recipient or person in whose behalf a message was received. See [MS-OXPROPS] for further explanation of the **PidTagRcvdRepresenting** message properties.

PidTagSender_XXX: Refers to **(PidTagSenderName, PidTagSenderEmailType and PidTagSenderEmailAddress),** or **PidTagSenderEntryId**, where either would suffice to fully represent the display name, e-mail transport type, and e-mail address of a sender or

person in whose behalf a message was sent. See [MS-OXPROPS] for further explanation of the **PidTagSender** message properties.

PidTagSentRepresenting_XXX: Refers to **(PidTagSentRepresentingName, PidTagSentRepresentingEmailType and PidTagSentRepresentingEmailAddress),** or **PidTagSentRepresentingEntryId**, where either would suffice to fully represent the display name, e-mail transport type, and e-mail address of a sender or person in whose behalf a message was sent. See [MS-OXPROPS] for further explanation of the **PidTagSentRepresenting** message properties.

2.2 ABNF Description

TNEFStream = TNEFHeader TNEFVersion OEMCodePage MessageData *AttachData

TNEFVersion = attrLevelMessage idTnefVersion Length TNEFVersionData Checksum

OEMCodePage = attrLevelMessage idOEMCodePage Length OEMCodePageData Checksum

MessageData = *MessageAttribute [MessageProps]
MessageAttribute = attrLevelMessage idMessageAttr Length Data Checksum
MessageProps = attrLevelMessage idMsgProps Length Data Checksum

; An attachment is determined/delimited by attAttachRendData, followed by
; other encoded attributes, if any and ending with attAttachment if there are
; any encoded properties

AttachData = AttachRendData [*AttachAttribute] [AttachProps]
AttachRendData = attrLevelAttachment idAttachRendData Length Data Checksum
AttachAttribute = attrLevelAttachment idAttachAttr Length Data Checksum
AttachProps = attrLevelAttachment idAttachment Length Data Checksum

; TNEF Version. TNEF Writers MUST use %x00.00.01.00
; TNEF Readers MUST reject other values
TNEFVersionData = 4*4OCTET

; This is the code page of attribute strings. MUST contain an 8-bit
; code page for compatibility with legacy applications that cannot handle
; “strings” with “embedded zero characters”.
OEMCodePageData = PrimaryCodePage SecondaryCodePage
PrimaryCodePage=UINT16
; Secondary CodePage is unused. SHOULD contain zero.
SecondaryCodePage=%x00.00

TNEFHeader = TNEFSignature LegacyKey

```
TNEFSignature = %x78.9F.3E.22

; Any number will suffice here. This is now legacy.
LegacyKey = UINT16

; The length of the following data field in bytes. All attribute lengths are
; 32-bit integers, including any terminating null characters.

Length = INT32

; Data of the attribute itself, flattened out based on the particular attribute
; according to the rules below

Data = 0*OCTET

; 16-bit unsigned integer that is the sum, modulo 65536, of the data bytes for
; the attribute value data, calculated over the entire length of the attribute data.
; In the case where the attribute contains enhanced properties with padding, the
; pad bytes MUST be included in the calculation.

Checksum = UINT16

; Level where attribute applies, either to the message itself or to
; an attachment

attrLevelMessage = %x01
attrLevelAttachment = %x02

; Attribute ID Tags

;TNEF Version
idTnefVersion = %x06.90.08.00

;OEM Codepage. See attOemCodepage handling in section 5 and Appendix A
idOemCodepage = %x07.90.06.00

; Message-level attributes. SHOULD all be at attrLevelMessage
idMessageAttr = idMessageClass / idFrom / idSubject / idDateSent /
  idDateRecd / idMessageStatus / idMessageID / idParentID /
  idConversationID / idBody / idPriority / idDateModified /
  idRecipTable / idOriginalMessageClass / idOwner / idSentFor /
  idDelegate / idDateStart / idDateEnd / idAidOwner / idRequestRes

; PidTagMessageClass
idMessageClass = %x08.80.07.00
```

; PidTagSenderEntryId
idFrom = %x00.80.00.00

; PidTagSubject
idSubject = %x04.80.01.00

; PidTagClientSubmitTime
idDateSent = %x05.80.03.00

; PidTagMessageDeliveryTime
idDateRecd = %x06.80.03.00

; PidTagMessageFlags
idMessageStatus = %x07.80.06.00

; PidTagSearchKey
idMessageID = %x09.80.01.00

; PidTagParentKey
idParentID = %x0A.80.01.00

; PidTagConversationKey
idConversationID = %x0B.80.01.00

; PidTagBody
idBody = %x0C.80.02.00

; PidTagImportance
idPriority = %x0D.80.x04.00

; PidTagLastModificationTime (message)
idDateModified = %x20.80.03.00

; Message Property Encapsulation
idMsgProps = %x03.90.06.00

; PidTagMessageRecipients. See attRecipTable handling
; in section 5.
idRecipTable = %x04.90.06.00

; PidTagOrigMessageClass
idOriginalMessageClass = %x00.06.07.00

; PidTagRcvdRepresentingEmailAddress or PidTagSentRepresentingEmailAddress

idOwner = %x00.00.06.00

; PidTagSentRepresentingEmailAddress
idSentFor = %x00.01.06.00

; PidTagRcvdRepresentingEmailAddress
idDelegate = %x00.02.06.00

; PidTagStartDate
idDateStart = %x00.06.03.00

; PidTagEndDate
idDateEnd = %x00.07.03.00

; PidTagOwnerApptId
idAidOwner = %x00.08.05.00

; PidTagResponseRequested
idRequestRes = %x00.09.04.00

; Attachment-level attributes. MUST all be at attrLevelAttachment
idAttachAttr = idAttachData / idAttachTitle / idAttachMetaFile /
idAttachCreateDate / idAttachModifyDate / idAttachTransportFilename

; PidTagAttachDataBin or PidTagAttachDataObj
idAttachData = %x0F.80.06.00

; PidTagAttachFilename
idAttachTitle = %x10.80.01.00

; PidTagAttachRendering
idAttachMetaFile = %x11.80.06.00

; PidTagCreationTime
idAttachCreateDate = %x12.80.03.00

; PidTagLastModificationTime (attachment)
idAttachModifyDate = %x13.80.03.00

; PidTagAttachTransportName
idAttachTransportFilename = %x01.90.06.00

;Attachment RendData
idAttachRendData = %x02.02.06.00

; Attachment table Row
idAttachment = %x05.90.06.00

2.3 Attributes

This section describes the attributes that appear in the TNEF stream, including the structure of the attributes, the message properties they map to, and any required conversions between them and message properties.

Each attribute is described by a level at which it applies (message or attachment), the attribute id, the length of the attribute data, the attribute data and a simple checksum. They are documented below in the form "**attXXXX**", for example attSubject refers to:

*attrLevelMessage **idSubject** Length *VCHAR Checksum*

2.3.1 attTnefVersion

Originally meant to permit versioning of the structure, this attribute is now legacy. TNEF Writers MUST write it as %x00.00.01.00 and TNEF readers MUST reject other content.

2.3.2 attOemCodepage

Contains the Windows Codepage used by the TNEF Writer for all attribute string values, and for any ANSI strings in the encapsulated message properties. See section 5.1 attOemCodepage handling in section 5 for a discussion of codepage handling.

2.3.3 attFrom

Level=Message. Maps to/from: **PidTagSender_XXX**

The data for this attribute encodes as follows:

TRP-structure = TRP-header sender-display-name sender-email

TRP-header = trpidOneOff structure-length sender-name-length sender-email-length

; Structure type
trpidOneOff = %x04.00

; The length of the entire structure. See description below.
structure-length = UINT16

; Length of sender name string, including the terminating zero character
sender-name-length = UINT16

; Length of sender email string, including the terminating zero character
sender-email-length = UINT16

sender-display-name = 1*OCTET %x00

sender-email = sender-email-type ":" sender-email-address %x00

sender-email-type = 1*CHAR

sender-email-address = 1*CHAR

The structure-length field is calculated as 8 (the size of TRP-header in OCTETs) plus the length of sender-display-name (including the terminating zero character), and the length of sender-email (including the terminating zero character).

The sender-name-length field is the length of sender-display-name in OCTETs (including the terminating zero character).

The sender-email-length field is calculated as the length of sender-email (including the terminating zero character)

sender-email is a string that is composed of four parts, the address-type (for example, the literal sequence "SMTP" for Internet addresses), a literal colon (":"), the address itself, and a terminating zero character. For example, the string *"SMTP:user2@example.com" %x00* is a legal sender-email value.

TNEF Writers MAY use the discrete **PidTagSender Name**, **AddressType** and **Address** properties if present, or access their values using the **PidTagSenderEntryId** property; TNEF Readers MUST set the **PidTagSenderEntryId** property and SHOULD decode the attOwner attribute into the other **PidTagSender** properties. For details of the EntryID property structure, see [MS-OXCMSG]

2.3.4 Date attributes

The data for these attributes encode into a Date Time Record structure, as follows:

DTR = wYear wMonth wDay wHour wMinute wSecond wDayOfWeek

wYear = UINT16
wMonth = UINT16
wDay = UINT16
wHour = UINT16
wMinute = UINT16
wSecond = UINT16
wDayOfWeek = UINT16

wYear contains the year (2008 would be %xD8.07); wMonth is 1 for January and so on; wDay is 1 for the first day of the month; wHour, wMinute and wSecond contain the time; wDayOfWeek is 1 for Monday.

Attribute (in TNEF)	*Level*	*Message Property*
attDateSent	Message	PidTagClientSubmitTime
attDateRecd	Message	PidTagMessageDeliveryTime
attAttachCreateDate	Attachment	PidTagCreationTime
attAttachModifyDate	Attachment	PidTagLastModificationTime
attDateModified	Message	PidTagLastModificationTime
attDateStart	Message	PidTagStartDate
attDateEnd	Message	PidTagEndDate

2.3.5 Message Class attributes

The message class attribute value is stored as a zero-terminated string which usually will hold the name specified by the client.

- TNEF Writers MAY represent any message class whose value is shown in the "Message Property Value" column of the following table using the value in the "Attribute Value" column.
- TNEF Readers MUST do the appropriate reverse mapping: if attMessageClass contains a value that is shown in the "Attribute Value" column, the value that MUST be returned is that of the value in the "Message Property Value" column. If the attMessageClass value begins with the string "Microsoft Mail v3.0 ", the TNEF Reader MUST ignore the prefix while checking for the specially mapped message class values.
- TNEF Readers SHOULD ignore the checksum value for message class attributes because some legacy TNEF Writers generated invalid checksum values.
- Other values SHOULD be written and read in their original form.

Attribute Value (in TNEF)	*Message Property Value*
IPM.Microsoft Mail.Note	IPM.Note
IPM.Microsoft Mail.Read Receipt	Report.IPM.Note.IPNRN
IPM.Microsoft Mail.Non-Delivery	Report.IPM.Note.NDR
IPM.Microsoft Schedule.MtgRespP	IPM.Schedule.Meeting.Resp.Pos
IPM.Microsoft Schedule.MtgRespN	IPM.Schedule.Meeting.Resp.Neg
IPM.Microsoft Schedule.MtgRespA	IPM.Schedule.Meeting.Resp.Tent
IPM.Microsoft Schedule.MtgReq	IPM.Schedule.Meeting.Request
IPM.Microsoft Schedule.MtgCncl	IPM.Schedule.Meeting.Canceled

Message class attributes in TNEF are as follows:

Attribute (in TNEF)	*Level*	*Message Property*
attMessageClass	Message	PidTagMessageClass

attOriginalMessageClass	Message	PidTagOrigMessageClass

2.3.6 Conversation-tracking attributes

These are stored by clients using binary, and stored in the attribute in text format. For compatibility, when generating a TNEF structure, the TNEF Writer MUST translate the binary representation into a textual one by emitting the two CHAR hexadecimal equivalent per OCTET (i.e. %x01 becomes "01", %x2D becomes "2D", etc). Likewise, when reading the attribute from TNEF, the TNEF Reader SHOULD attempt to decode the text format back into binary.

Attribute (in TNEF)	*Level*	*Message Property*
attMessageID	Message	PidTagSearchKey
attParentID	Message	PidTagParentKey
attConversationID	Message	PidTagConversationKey

2.3.7 attSubject

Level=Message. Maps to/from: **PidTagSubject**

The data for this attribute is stored as a string with a terminating zero character.

2.3.8 attMessageStatus

Level=Message. Maps to/from: **PidTagMessageFlags**

The data for this attribute is stored as an unsigned 32-bit integer, with the appropriate bits set to indicate the message status. For compatibility, these MUST be mapped to/from values:

Status	*Attribute Flag (in TNEF)*	*Bit Value*	*Message Property Flag*	*Bit Value*
Read	fmsRead	0x20	MSGFLAG_READ	0x01
Unmodified	fmsModified	0x01	MSGFLAG_UNMODIFIED	0x02
Submitted	fmsSubmitted	0x04	MSGFLAG_SUBMIT	0x04
Unsent	fmsLocal	0x02	MSGFLAG_UNSENT	0x08
Has Attachments	fmsHasAttach	0x80	MSGFLAG_HASATTACH	0x10

The states of the bit values match except for the "unmodified" flag. The value of fmsModified equals the negative of MSGFLAG_UNMODIFIED.

2.3.9 attBody

Level=Message. Maps to/from: **PidTagBody**

The data for this attribute is stored as a string with a terminating zero character.

2.3.10 attPriority

Level=Message. Maps to/from: **PidTagImportance**

The data for this attribute is stored as an unsigned 16-bit integer. For compatibility, these MUST be mapped to/from values:

Priority	*Value (in TNEF)*	*Message Property Value*
Low	3	0
Normal	2	1
High	1	2

2.3.11 attAttachData

Level=Attachment. Maps to/from: **PidTagAttachDataBin**

The data for this attribute is stored as a binary stream.

TNEF Writers MAY use this attribute for the attachment data if **PidTagAttachMethod** of the original attachment contains the value 0x0001 (ATTACH_BY_VALUE). For other values, the data MUST be encoded in **attAttachment**.

TNEF Writers MAY choose to put alternate text (such as "This is an attached message") in attAttachData for other cases. TNEF Readers MAY take advantage of this text for lightweight message display, but SHOULD use the data from attAttachment when doing a full decode.

2.3.12 attAttachTitle

Level=Attachment. Maps to/from: **PidTagAttachFileName**

The data for this attribute is stored as a string with a terminating zero character.

2.3.13 attAttachMetaFile

Level=Attachment. Maps to/from: **PidTagAttachRendering**

The data for this attribute is stored as a binary stream. See [MS-WMF] for a description of the content.

2.3.14 attAttachTransportFilename

Level=Attachment. Maps to/from: **PidTagAttachTransportName**

The data for this attribute is stored as a string with a terminating zero character.

2.3.15 attAttachRendData

This attachment-level attribute contains a structure containing information that can be used for rendering the attachment in the body.

Each set of attachment attributes MUST begin with attAttachRendData, followed by any other attributes; attachment properties encoded in attAttachment SHOULD be last.

The structure is laid out as follows:

attAttachRendData = AttachType AttachPosition RenderWidth RenderHeight DataFlags

AttachType = AttachTypeNull / AttachTypeFile / AttachTypeOle / AttachTypePicture

AttachTypeFile=%x02.00
AttachTypeOle=%x03.00

AttachPosition= INT32
RenderWidth=INT16
RenderHeight=INT16

DataFlags = FileDataDefault / FileDataMacBinary

FileDataDefault= %x00.00.00.00
FileDataMacBinary=%x01.00.00.00

AttachType MUST BE set by the TNEF Writer to *AttachTypeFile* when **PidTagAttachMethod** for the attachment is ATTACH_BY_VALUE or ATTACH_EMBEDDED_MSG, and to *AttachTypeOle* when **PidTagAttachMethod** is ATTACH_OLE.

TNEF Readers SHOULD set **PidTagAttachTag** for the attachment to %x2A.86.48.86.F7.14.03.0A.03.01.01 when *AttachTypeOle* is set.

AttachPosition maps to/from **PidTagRenderingPosition**

RenderWidth and **RenderHeight** MAY be set by the TNEF Writer to the size of the system icons if the attachment is a simple file attachment, or {-1,-1}. The messaging client SHOULD use {-1,-1} as a signal to compute them if needed for inline display of the attachment.

FileDataMacBinary SHOULD be set by the TNEF Writer if **PidTagAttachEncoding** for the attachment consists of the following bytes: %x2A.86.48.86.F7.14.03.0B.01, and the TNEF Reader SHOULD set the value of the **PidTagAttachEncoding** property for the attachment to the same bytes if *DataFlags* contains *FileDataMacBinary*.

2.3.16 attOwner

Level=Message. Meeting attribute.
Maps to/from: **PidTagRcvdRepresenting_XXX** or **PidTagSentRepresenting_XXX**

The attOwner attribute is encoded as counted strings laid end-to-end. The format for attOwner is as follows:

attOwner=display-name-length display-name address-length address

; Length of "display-name" including terminating zero character
display-name-length=UINT16
display-name=*CHAR %x00

;Length of "address" including terminating zero character
address-length=UINT16
address= 1*CHAR ":" 1*CHAR %x00

The display-name-length and address-length are unsigned 16-bit values, including the terminating zero characters for the strings. The type and address strings in the email-address entry are separated by a colon (:) character, e.g. "SMTP:user1@example.com" %x00.

The mapping of message properties to the attOwner attribute is dependent on the message class of the message being encoded.

If the message is either a Meeting Request or Meeting Cancellation (from organizer, creating or deleting a meeting), then the TNEF Writer SHOULD encode the **PidTagSentRepresenting** properties in the **attOwner** attribute and the TNEF Reader SHOULD decode the **attOwner** attribute into the **PidTagSentRepresenting** properties.

If the message is a meeting response of any type (from attendee, accepting, declining, etc), the TNEF Writer SHOULD encode the **PidTagRcvdRepresenting** properties in the **attOwner** attribute and the TNEF Reader SHOULD decode the attOwner attribute into the **PidTagRcvdRepresenting** properties. For details on how to construct the EntryID property, see [MS-OXCMSG].

TNEF Writers MAY use the discrete **PidTag{Sent | Rcvd}Representing Name**, **AddressType** and **Address** properties if present, or access their values using the **PidTag{Sent | Rcvd}EntryId** property; TNEF Readers MUST set the **PidTag{Sent | Rcvd} EntryId** property and SHOULD decode the attOwner attribute into the other **PidTag{Sent | Rcvd} Representing** properties .

2.3.17 attSentFor

Level=Message. Meeting attribute. Maps to/from: **PidTagSentRepresenting_XXX**

The TNEF Writer SHOULD encode the **PidTagSentRepresenting** properties in the **attSentFor** attribute and the TNEF Reader SHOULD decode the **attSentFor** attribute into the **PidTagSentRepresenting** properties.

TNEF Writers MAY use the discrete **PidTagSentRepresenting Name**, **AddressType** and **Address** properties if present, or access their values using the **PidTagSentRepresentingEntryId** property; TNEF Readers SHOULD set the **PidTagSentRepresentingEntryId** property and SHOULD decode the attOwner attribute into the other **PidTagSentRepresenting** properties.

2.3.18 attDelegate

Level=Message. Meeting attribute. Maps to/from: **PidTagDelegation**

The content can be transported as a binary large object (BLOB). See [MS-OXCAL] for an explanation of the **PidTagDelegation** property.

2.3.19 attAidOwner

Level=Message. Meeting attribute. Maps to/from: **PidTagOwnerApptId**

The content can be transported as a BLOB. See [MS-OXCAL] for an explanation of the **PidTagOwnerApptId** property.

2.3.20 attRequestRes

Level=Message. Meeting attribute. Maps to/from: **PidTagResponseRequested**

This is a Boolean; it SHOULD be transported as a 16-bit integer (the low order 16 bits of PidTagResponseRequested). See [MS-OXCAL] for an explanation of the **PidTagResponseRequested** property.

2.3.21 attMsgProps

Level=Message. Maps to/from an arbitrary set of message properties.

TNEF Writers SHOULD write this attribute after all of the other message attributes and immediately before all of the attachment attributes. See “Encapsulated Message Properties” for a description of the encoding for this attribute.

2.3.22 attRecipTable

Level=Message. Maps to/from: **PidTagMessageRecipients**

See “Encapsulated Message Properties” for a description of the encoding for this attribute.

2.3.23 attAttachment

Level=Attachment. Maps to/from an arbitrary set of properties for a single attachment.

TNEF Writers SHOULD write this encapsulation after all of the attributes for an attachment. See "Encapsulated Message Properties" for a description of the encoding for this attribute.

2.4 Encapsulated Message Properties

The following attributes are special in that they are used to encode any message property that does not have a counterpart in the set of existing TNEF-defined attributes. The attribute data is a counted set of message properties laid end-to-end. The format of this encoding, which allows for any set of message properties, is as follows:

```
MsgPropertyList = 1*MsgPropertyCount *MsgPropertyValue
MsgPropertyCount = UINT32

MsgPropertyValue = MsgPropertyTag MsgPropertyContent
MsgPropertyTag = MsgPropertyType MsgPropertyId [NamedPropSpec]

; Only present when MsgPropertyId is > 0x8000
NamedPropSpec = PropNameSpace PropIDType PropMap

; Contains a GUID to specify the namespace
PropNameSpace = GUID
PropIDType = IDTypeNumber / IDTypeString
PropMap = PropMapID / PropMapString

IDTypeNumber = %x00.00.00.00
IDTypeString = %x01.00.00.00

; Used if PropIDType is IDTypeNumber. Contains a number
; that is used to identify the property within the namespace
PropMapID = UINT32

; Used if PropIDType is IDTypeString. Contains a length, then a
; UTF-16LE encoded UNICODE string
; and terminating 2-byte zero character, padded to UINT32 boundary
PropMapString =  UINT32 *UINT16 %00.00 [PropMapPad]

;Padding for a UTF-16LE encoded UNICODE string to UINT32 size.
;Should be either 0 or 2 bytes
PropMapPad=*1UINT16

MsgPropertyType = TypeUnspecified / TypeNull / TypeInt16 / TypeInt32 /
 TypeFlt32 / TypeFlt64 / TypeCurrency / TypeAppTime / TypeError /
 TypeBoolean /  TypeObject / TypeInt64 / TypeString8 / TypeUnicode /
 TypeSystime / TypeCLSID / TypeBinary / TypeMVInt16 / TypeMVInt32 /
 TypeMVFlt32 / TypeMVFlt64 / TypeMVCurrency / TypeMVAppTime /
 TypeMVSystime / TypeMVString8 / TypeMVBinary / TypeMVUnicode /
```

TypeMVCLSID / TypeMVInt64

MsgPropertyID = UINT16

; This is used on a calling interface, MUST NOT be on a stored property
TypeUnspecified = %x00.00

; This is returned from a calling interface, MUST NOT be on a stored property
TypeNull = %x01.00

; Signed 16-bit value = INT16
TypeInt16 = %x02.00
TypeMVInt16 = %x02.10

; Signed 32-bit value = INT32
TypeInt32 = %x03.00
TypeMVInt32 = %x03.10

; Signed 32-bit floating point= FLOAT
TypeFlt32 = %x04.00
TypeMVFlt32 = %x04.10

; 64-bit floating point= DOUBLE
TypeFlt64 = %x05.00
TypeMVFlt64 = %x05.10

; Signed 64-bit int = OLE CURRENCY type
TypeCurrency = %x06.00
TypeMVCurrency = %x06.10

; Application time= OLE DATE type
TypeAppTime = %x07.00
TypeMVAppTime = %x07.10

; This is returned from a calling interface, MUST NOT be on a stored property
TypeError = %x0A.00

; 16-bit Boolean (non-zero = true)
TypeBoolean = %x0B.00

; Embedded object on a property
TypeObject = %x0D.00

; 8-byte signed integer= INT64
TypeInt64 = %x14.00

```
TypeMVInt64 = %x14.10

; 8-bit character string with terminating zero character
; See section 5.1 attOemCodepage handling Appendix A for more
; information about encoding and decoding 8-bit strings in TNEF
TypeString8 = %x1E.00
TypeMVString8 = %x1E.10

; UTF-16LE or variant character string with terminating 2-byte zero character
TypeUnicode = %x1F.00
TypeMVUnicode = %x1F.10

; FILETIME
TypeSystime = %x40.00
TypeMVSystime = %x40.10

; OLE GUID
TypeCLSID = %x48.00
TypeMVCLSID = %x48.10

; Uninterpreted binary BLOB
TypeBinary = %x02.01
TypeMVBinary = %x02.11

; MsgPropertyContent varies by property type.
MsgPropertyContent = PropertyScalarContent / PropertyMultiScalarContent /
                     PropertyMultiVariableContent

; Scalars – Types Int16, Int32, Flt32, Flt64, Currency, AppTime,
;           Bool, Int64, Systime, CLSID
; The data for the particular property is written to the stream and if need
; be, padded with bytes (which SHOULD be zero) to achieve a multiple of 4-bytes in length

PropertyScalarContent = MsgPropertyContent [PropertyPad]

; PropertyPad – between 0 and 3 zero-filled bytes, added to the streamed
; property values to achieve 4-byte boundary. Writers MUST use zero bytes
; and Readers MUST permit non-zero bytes. These pad bytes MUST be counted
; in the checksum of the containing attribute.

PropertyPad=*3ZERO
ZERO= %x00

; Multi-value Scalars – Types MVInt16, MVInt32, MVFlt32, MVFlt64, MVCurrency,
;                        MVAppTime, MVBool, MVInt64, MVSystime, MVCLSID
```

; The number of values for the property is written to the stream as a 4 byte
; value, then the data for each value is written to the stream and if need
; be, padded with bytes (which SHOULD be zero) to achieve a multiple of 4-bytes in length

PropertyMultiScalarContent = PropertyContentCount *MsgPropertyContent [PropertyPad]
PropertyContentCount = UINT32

; Variable-length – Types Unicode, String8, Object, Binary
;These are handled as a special case of Multi-Variable-Length with the number of values=1.

; Multi-Variable-length – Types MVUnicode, MVString8, MVBinary
; The number of values for the property is written to the stream as a 4 byte value,
; then for each value, the size of the property is written to the stream as a 4 byte
; value, then the data for the property is written to the stream and if need be, padded
; with zero bytes to achieve a multiple of 4-bytes in length. No padding is necessary
; since each value was itself padded to a 4-byte boundary.

PropertyMultiVariableContent = MsgPropertyCount 1*PropertyVariableContent
MsgPropertyCount = UINT32

; The size of the property is written to the stream as a 4 byte value, then the data
; for the property is written to the stream and if need be, padded with zero bytes
; to achieve a multiple of 4-bytes in length. The size includes the interface
; identifier at the beginning of the value stream for an object but does not include
; the padding bytes.
;
; If the property is of type PT_OBJECT, the value-size is followed by the interface
; identifier of the object, then the serialized stream of the object's data. Only the
; interface identifiers of the OLE IStorage, the OLE IStream, and the
; Outlook/Exchange IMessage object are supported. The size of the interface
; identifier is included in the calculation of value-size.
;
; If the object is an attached message (that is, it has a property type of PT_OBJECT
; and an interface identifier %x07.03.02.00.00.00.00.00.C0.00.00.00.00.00.00.46,
; that of a message object), the value data is encoded as an
; embedded TNEF stream.

PropertyVariableContent = MsgPropertySize MsgPropertyValue [PropertyPad]
MsgPropertySize = UINT32

;Attributes containing encapsulated properties:

;attMsgProps - Maps to/from an arbitrary set of message properties
MsgPropsData = MsgPropertyList

;attRecipTable - Maps to/from PidTagMessageRecipients
;Number of table rows followed by the rows
RecipTableData = NumRows *RecipRow

NumRows = UINT32
RecipRow = MsgPropertyList

attAttachment – maps to/from an arbitrary set of properties for a single attachment.
AttachPropsData = MsgPropertyList

3 Structure Examples

3.1 Sample Message

This message was sent within an Exchange organization.

Explanation of content	*Byte stream*
Tnef Signature, Value=0x223E9F78	78 9f 3e 22
Attach Key, Value=0x0001	01 00
Message attribute, attTnefVersion Length=4 bytes	01 06 90 08 00 04 00 00 00
Data=0x00000001	00 00 01 00
Checksum=0x0001	01 00
Message attribute, attOemCodepage Length=8 bytes	01 07 90 06 00 08 00 00 00
Data= (Primary=1252, Secondary=0)	e4 04 00 00 00 00 00 00
Checksum=0x00E8	e8 00
Message attribute, attMessageClass Length=24 bytes	01 08 80 07 00 20 00 00 00
Data="IPM.Microsoft Mail.Note" %x00	49 50 4d 2e 4d 69 63 72 6f 73 6f 66 74 20 4d 61 69 6c 2e 4e 6f 74 65 00
Checksum=0x0831	31 08
Message attribute, attPriority Length=2 bytes	01 0d 80 04 00 02 00 00 00
Data=0x0002 (Normal Priority)	02 00
Checksum=0x0002	02 00
Message attribute, attSubject Length=15 bytes	01 04 80 01 00 0f 00 00 00
Data="Simple subject" %x00	53 69 6d 70 6c 65 20 73 75 62 6a 65 63 74 00
Checksum=0x0831	31 08
Message attribute, attDateSent Length=14 bytes	01 05 80 03 00 0e 00 00 00
Data= Tuesday, 02/17/2004, 11:25:35	d4 07 02 00 11 00 0b 00 19 00 23 00 02 00
Checksum=0x0137	37 01
Message attribute, attDateModified Length=14 bytes	01 20 80 03 00 0e 00 00 00
Data= Tuesday, 02/17/2004, 12:26:09	D4 07 02 00 11 00 0C 00 -- 1A 00 09 00 02 00
Checksum=0x011F	1F 01
Message attribute, attMessageID Length=33 bytes	01 09 80 01 00 21 00 00 00
Data= "757BB19CDE936A4087D90BB784C58E3B"	37 35 37 42 42 31 39 43 44 45 39 33 36 41 34 30 38 37 44 39 30 42 42 37 38 34 43 35 38 45 33 42 00
Checksum=0x0751	51 07
Message attribute, Encapsulation, attMsgProps Length=2256 bytes Number of properties=69	01 03 90 06 00 d0 08 00 00 45 00 00 00
Message property,	0b 00 02 00

PidTagAlternateRecipientAllowed Value=1 (TRUE)	01 00 00 00
Message property, PidTagPriority Value=0 (LOW)	03 00 26 00 00 00 00 00
Message property, PidTagSensitivity Value=0 (LOW)	03 00 36 00 00 00 00 00
Message property, PidTagClientSubmitTime Value=Tuesday, 02/17/2004, 19:25:35	40 00 39 00 AA D6 F3 D0 8B F5 C3 01
Message property, PidTagSubjectPrefix Number of property values=1 Size of first property=1	1e 00 3d 00 01 00 00 00 01 00 00 00
Value = %x00	00
Pad=3 bytes	00 00 00
Message property, PidTagReceivedByEntryid Number of property values=1 Size of first property=107	02 01 3f 00 01 00 00 00 6b 00 00 00
	00 00 00 00 DC A7 40 C8 C0 42 10 1A B4 B9 08 00 2B 2F E1 82 01 00 00 00 00 00 00 00 2F 4F 3D 46 49 52 53 54 20 4F 52 47 41 4E 49 5A 41 54 49 4F 4E 2F 4F 55 3D 46 49 52 53 54 20 41 44 4D 49 4E 49 53 54 52 41 54 49 56 45 20 47 52 4F 55 50 2F 43 4E 3D 52 45 43 49 50 49 45 4E 54 53 2F 43 4E 3D 54 45 53 54 32 31 55 57 32 00
Pad=1 byte	00
Message property, PidTagReceivedByName Number of property values=1 Size of first property=10	1e 00 40 00 01 00 00 00 0a 00 00 00
Value="Test21uw2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, **PidTagRcvdRepresentingEntryId** Number of property values=1 Size of first property=107	02 01 43 00 01 00 00 00 6b 00 00 00
	00 00 00 00 DC A7 40 C8 C0 42 10 1A B4 B9 08 00 2B 2F E1 82 01 00 00 00 00 00 00 00 2F 4F 3D 46 49 52 53 54 20 4F 52 47 41 4E 49 5A 41 54 49 4F 4E 2F 4F 55 3D 46 49 52 53 54 20 41 44 4D 49 4E 49 53 54 52 41 54 49 56 45 20 47 52 4F 55 50 2F 43 4E 3D 52 45 43 49 50 49 45 4E 54 53 2F 43 4E 3D 54 45 53 54 32 31 55 57 32 00
Pad=1 byte	00
Message property, **PidTagRcvdRepresentingName** Number of property values=1 Size of first property=10	1e 00 44 00 01 00 00 00 0a 00 00 00
Value="Test21uw2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, PidTagMessageSubmissionId Number of property values=1 Size of first property=57	02 01 47 00 01 00 00 00 39 00 00 00
	63 3D 55 53 3B 61 3D 20 3B 70 3D 46 69 72 73 74 20 4F 72 67 61 6E 69 7A 61 74 69 3B 6C 3D 4A 45 53 45 4F 47 50 55 57 32 2D 30 34 30 32 31 37 31 39 32 35 33 35 5A 2D 32 00
Pad=3 bytes	00 00 00
Message property, **PidTagReceivedBySearchKey** Number of property values=1 Size of first property=82	02 01 51 00 01 00 00 00 52 00 00 00
	45 58 3A 2F 4F 3D 46 49 52 53 54 20 4F 52 47 41 4E 49 5A 41 54 49 4F 4E 2F 4F 55 3D 46 49 52 53 54 20 41 44 4D 49 4E 49 53 54 52 41 54 49 56 45 20 47 52 4F 55 50 2F 43 4E 3D 52 45 43 49 50 49 45 4E 54 53 2F 43 4E 3D 54 45 53 54 32 31 55 57 32 00
Pad=2 bytes	00 00
Message property,	02 01 52 00

PidTagRcvdRepresentingSearchKey Number of property values=1 Size of first property=82	01 00 00 00 52 00 00 00
	45 58 3A 2F 4F 3D 46 49 52 53 54 20 4F 52 47 41 4E 49 5A 41 54 49 4F 4E 2F 4F 55 3D 46 49 52 53 54 20 41 44 4D 49 4E 49 53 54 52 41 54 49 56 45 20 47 52 4F 55 50 2F 43 4E 3D 52 45 43 49 50 49 45 4E 54 53 2F 43 4E 3D 54 45 53 54 32 31 55 57 32 00
Pad=2 bytes	00 00
Message property, PidTagMessageToMe Value=1 (TRUE)	00 0b 00 57 00 01 00 00
Message property, PidTagMessageCcMe Value=0 (FALSE)	00 0b 00 58 00 00 00 00
Message property, PidTagMessageRecipMe Value=1 (TRUE)	00 0b 00 59 00 01 00 00
Message property, PidTagConversationTopic Number of property values=1 Size of first property=15	1e 00 70 00 01 00 00 00 0f 00 00 00
Value=”Simple subject” %x00	53 69 6D 70 6C 65 20 73 75 62 6A 65 63 74 00
Pad=1 byte	00
Message property, PidTagConversationIndex Number of property values=1 Size of first property=22	02 01 71 00 01 00 00 00 16 00 00 00
	01 C3 F5 8B 07 63 76 8D 89 1B D0 79 47 C9 8F 66 4E 21 9A D2 4A F2
Pad=2 bytes	00 00
Message property, PidTagReceivedByAddrtype Number of property values=1 Size of first property=5	1e 00 75 00 01 00 00 00 05 00 00 00
Value=”SMTP” %x00	53 4d 54 50 00
Pad=3 bytes	00 00 00
Message property, **PidTagReceivedByEmailAddress** Number of property values=1 Size of first property=40	1e 00 76 00 01 00 00 00 28 00 00 00
Value=”Test21uw2@mydomuw2.extest.microsoft.com” %x00	54 65 73 74 32 31 75 77 32 40 6D 79 64 6F 6D 75 77 32 2E 65 78 74 65 73 74 2E 6D 69 63 72 6F 73 6F 66 74 2E 63 6F 6D 00
Message property, **PidTagRcvdRepresentingAddrtype** Number of property values=1 Size of first property=5	1e 00 77 00 01 00 00 00 05 00 00 00
Value=”SMTP” %x00	53 4d 54 50 00
Pad=3 bytes	00 00 00
Message property, **PidTagRcvdRepresentingEmailAddress** Number of property values=1 Size of first property=40	1e 00 78 00 01 00 00 00 28 00 00 00
Value=”Test21uw2@mydomuw2.extest.microsoft.com” %x00	54 65 73 74 32 31 75 77 32 40 6D 79 64 6F 6D 75 77 32 2E 65 78 74 65 73 74 2E 6D 69 63 72 6F 73 6F 66 74 2E 63 6F 6D 00
Message property, PidTagSenderName Number of property values=1 Size of first property=10	1e 00 1A 0C 01 00 00 00 0a 00 00 00
Value=”Test21uw2” %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, PidTagNormalizedSubject Number of property values=1 Size of first property=15	1e 00 1D 0E 01 00 00 00 0f 00 00 00
Value=”Simple subject” %x00	53 69 6D 70 6C 65 20 73 75 62 6A 65 63 74 00
Pad=1 byte	00
Message property, PidTagRtfCompressed Number of property values=1 Size of first property=150	02 01 09 10 01 00 00 00 96 00 00 00
	92 00 00 00 AB 00 00 00 4C 5A 46 75 A8 14 2F 17 03 00 0A 00 72 63 70 67 31 32 35 16 32 00 F8 0B

	60 6E 0E 10 30 33 33 4F 01 F7 02 A4 03 E3 02 00 63 68 0A C0 73 B0 65 74 30 20 07 13 02 80 7D 0A 80 9D 00 00 2A 09 B0 09 F0 04 90 61 74 05 B1 1A 52 0D E0 68 09 80 01 D0 20 35 2E 30 35 30 2E 33 13 B0 01 D0 30 32 49 02 80 5C 76 08 90 77 6B 0B 80 64 FA 34 0C 60 63 00 50 0B 03 0B B5 06 00 07 70 C9 0B 50 65 20 07 81 73 61 12 50 0A A2 0B 0A 80 11 E1 00 17 A0
Pad=2 bytes	00 00
Message property, PidTagInternetMessageId Number of property values=1 Size of first property=80	1e 00 35 10 01 00 00 00 50 00 00 00
Value="<2896107D7E52DF4DB5D10536D BFEFAD07E37@jeseogpuw2.mydomuw2. extest.microsoft.com>" %x00	3C 32 38 39 36 31 30 37 44 37 45 35 32 44 46 34 44 42 35 44 31 30 35 33 36 44 42 46 45 46 41 44 30 37 45 33 37 40 6A 65 73 65 6F 67 70 75 77 32 2E 6D 79 64 6F 6D 75 77 32 2E 65 78 74 65 73 74 2E 6D 69 63 72 6F 73 6F 66 74 2E 63 6F 6D 3E 00
Message property, PidTagIconIndex Value=0xFFFF	03 00 80 10 ff ff ff ff
Message property, PidTagImapCachedMsgsize Number of property values=1 Size of first property=40	02 01 f0 10 01 00 00 00 28 00 00 00
	03 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 08 00 00 00 DB 11 00 00 49 53 4F 2D 38 38 35 39 2D 31 00 00
Message property, PidTagUrlCompName Number of property values=1 Size of first property=38	1f 00 f3 10 01 00 00 00 26 00 00 00
Value="Simple subject.EML" %x00.00 (UNICODE)	53 00 69 00 6D 00 70 00 6C 00 65 00 20 00 73 00 75 00 62 00 6A 00 65 00 63 00 74 00 2E 00 45 00 4D 00 4C 00 00 00
Pad=2 bytes	00 00
Message property, PidTagAttrHidden Value=0 (FALSE)	0b 00 f4 10 00 00 00 00
Message property, PidTagAttrSystem Value=0 (FALSE)	0b 00 f5 10 00 00 00 00
Message property, PidTagAttrReadonly Value=0 (FALSE)	0b 00 f6 10 00 00 00 00
Message property, PidTagCreationTime Value=Tuesday, 02/17/2004, 20:26:09	40 00 07 30 90 24 46 47 94 f5 c3 01
Message property, **PidTagLastModificationTime** Value=Tuesday, 02/17/2004, 20:26:09	40 00 08 30 90 24 46 47 94 f5 c3 01
Message property, PidTagInternetCpid Value=(Primary=1252, Secondary=0)	03 00 de 3f e4 04 00 00
Message property, PidTagMessageLocaleId Value=(Primary=1033, Secondary=0)	03 00 f1 3f 09 04 00 00
Message property, PidTagCreatorName Number of property values=1 Size of first property=10	1e 00 f8 3f 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, PidTagCreatorEntryid Number of property values=1 Size of first property=107	02 01 F9 3F 01 00 00 00 6b 00 00 00
	00 00 00 00 DC A7 40 C8 C0 42 10 1A B4 B9 08 00 2B 2F E1 82 01 00 00 00 00 00 00 00 2F 4F 3D 46 49 52 53 54 20 4F 52 47 41 4E 49 5A 41 54 49 4F 4E 2F 4F 55 3D 46 49 52 53 54 20 41 44 4D 49 4E 49 53 54 52 41 54 49 56 45 20 47 52 4F 55 50 2F 43 4E 3D 52 45 43 49 50 49 45 4E 54 53 2F 43 4E 3D 54 45 53 54 32 31 55 57 32 00
Pad=1 byte	00
Message property, PidTagLastModifierName Number of property values=1 Size of first property=10	1e 00 fa 3f 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00

Pad=2 bytes	00 00
Message property, PidTagLastModifierEntryid Number of property values=1 Size of first property=107	02 01 FB 3F 01 00 00 00 6b 00 00 00
	00 00 00 00 DC A7 40 C8 C0 42 10 1A B4 B9 08 00 2B 2F E1 82 01 00 00 00 00 00 00 00 2F 4F 3D 46 49 52 53 54 20 4F 52 47 41 4E 49 5A 41 54 49 4F 4E 2F 4F 55 3D 46 49 52 53 54 20 41 44 4D 49 4E 49 53 54 52 41 54 49 56 45 20 47 52 4F 55 50 2F 43 4E 3D 52 45 43 49 50 49 45 4E 54 53 2F 43 4E 3D 54 45 53 54 32 31 55 57 32 00
Pad=1 byte	00
Message property, PidTagMessageCodepage Value=(Primary=1252, Secondary=0)	03 00 fd 3f e4 04 00 00
Message property, PidTagSenderFlags Value=0	03 00 19 40 00 00 00 00
Message property, **PidTagSentRepresentingFlags** Value=0	03 00 1A 40 00 00 00 00
Message property, PidTagRcvdByFlags Value=0	03 00 1B 40 00 00 00 00
Message property, **PidTagRcvdRepresentingFlags** Value=0	03 00 1C 40 00 00 00 00
Message property, **PidTagSenderSimpleDispName** Number of property values=1 Size of first property=10	1e 00 30 40 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, **PidTagSentRepresentingSimpleDispName** Number of property values=1 Size of first property=10	1e 00 31 40 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, **PidTagRcvdBySimpleDispName** Number of property values=1 Size of first property=10	1e 00 34 40 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, **PidTagRcvdRepresentingSimpleDispName** Number of property values=1 Size of first property=10	1e 00 35 40 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, **PidTagCreatorSimpleDispName** Number of property values=1 Size of first property=10	1e 00 38 40 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, **PidTagLastModifierSimpleDispName** Number of property values=1 Size of first property=10	1e 00 39 40 01 00 00 00 0a 00 00 00
Value="TEST21UW2" %x00	54 65 73 74 32 31 75 77 32 00
Pad=2 bytes	00 00
Message property, PidTagContentFilterScl Value=0xFFFFFFFF	03 00 76 40 ff ff ff ff
Message property, PidTagMsgEditorFormat Value=0x00000003	03 00 09 59 03 00 00 00
Message property, PidTagTnefCorrelationKey Number of property values=1 Size of first property=12	02 01 7f 00 01 00 00 00 0c 00 00 00

	38 71 6b 6a 30 30 73 67 6d 34 66 00
Message Property, PidLidAgingDontAgeMe PSETID_Common, MNID_ID, dispidAgingDontAgeMe (0x850E) Value=0	0b 00 87 81 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 0e 85 00 00 00 00 00 00
Message Property, PidLidCurrentVersion PSETID_Common, MNID_ID, dispidCurrentVersion (0x8552) Value=115608	03 00 9f 81 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 52 85 00 00 98 c3 01 00
Message Property, PidLidCurrentVersionName PSETID_Common, MNID_ID, dispidCurrentVersionName (0x8554) Number of property values=1 Size of first property=5	1E 00 A0 81 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 54 85 00 00 01 00 00 00 05 00 00 00
Value="11.0" %x00	31 31 2E 30 00
Pad=3 bytes	00 00 00
Message Property, PidLidReminderDelta PSETID_Common, MNID_ID, dispidReminderDelta (0x8501) Value=0	03 00 eb 81 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 01 85 00 00 00 00 00 00
Message Property, PidLidReminderSet PSETID_Common, MNID_ID, dispidReminderSet (0x8503) Value=0 (FALSE)	0b 00 f0 81 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 03 85 00 00 00 00 00 00
Message Property, PidLidSideEffects PSETID_Common, MNID_ID, dispidSideEffects (0x8510) Value=0	03 00 fa 81 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 10 85 00 00 00 00 00 00
Message Property, PidLidTaskMode PSETID_Common, MNID_ID, dispidTaskMode (0x8518) Value=0	03 00 01 82 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 18 85 00 00 00 00 00 00
Message Property, PidLidPrivate PSETID_Common, MNID_ID, dispidPrivate (0x8506) Value=0 (FALSE)	0b 00 43 82 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 06 85 00 00 00 00 00 00
Message Property, PidLidUseTNEF PSETID_Common, MNID_ID, dispidPrivate (0x8582) Value=0 (FALSE)	0b 00 44 82 08 20 06 00 00 00 00 00 c0 00 00 00 00 00 00 46 00 00 00 00 82 85 00 00 00 00 00 00
Message property, **PidTagReadReceiptRequested** Value=0 (FALSE)	0b 00 29 00 00 00 00 00
Message property, **PidTagOriginatorDeliveryReportRequested** Value=0 (FALSE)	0b 00 23 00 00 00 00 00
Message property, PidTagRtfSyncBodyCRC Value=0xF00D29FF	03 00 06 10 ff 29 0d f0
Message property, PidTagRtfSyncBodyCount Value=13	03 00 07 10 0d 00 00 00
Message property, PidTagRtfSyncPrefixCount Value=0	03 00 10 10 00 00 00 00
Message property, **PidTagRtfSyncTrailingCount** Value=0	03 00 11 10 00 00 00 00
Message property, PidTagRtfSyncBodyTag Number of property values=1 Size of first property=14	1e 00 08 10 01 00 00 00 0e 00 00 00
Value="SIMPLEMESSAGE" %x00	53 49 4d 50 4c 45 4d 45 53 53 41 47 45 00

Pad=2 bytes	00 00
Message property, PidTagTnefCorrelationKey Number of property values=1 Size of first property=56	02 01 7f 00 01 00 00 00 38 00 00 00
	3C 32 38 39 36 31 30 37 44 37 45 35 32 44 46 34 44 42 35 44 31 30 35 33 36 44 42 46 45 46 41 44 30 37 45 33 37 40 75 73 65 72 2E 65 78 61 6D 70 6C 65 2E 63 6F 6D 3E 00
Checksum for **AttMsgProps**	45 C1

3.2 Sample Meeting Response

This example is particularly short, as it was sent by a mobile application.

Explanation of content	*Byte stream*
Tnef Signature, Value=0x223E9F78	78 9f 3e 22
Attach Key, Value=0x0001	01 00
Message attribute, attTnefVersion Length=4 bytes	01 06 90 08 00 04 00 00 00
Data=0x00000001	00 00 01 00
Checksum=0x0001	01 00
Message attribute, attOemCodepage Length=8 bytes	01 07 90 06 00 08 00 00 00
Data= (Primary=1252, Secondary=0)	e4 04 00 00 00 00 00 00
Checksum=0x00E8	e8 00
Message attribute, attMessageClass Length=32 bytes	01 08 80 07 00 20 00 00 00
Data="IPM.Microsoft Schedule.MtgRespN" %x00	49 50 4d 2e 4d 69 63 72 6f 73 6f 66 74 20 53 63 68 65 64 75 6c 65 2e 4d 74 67 52 65 73 70 4e 00
Checksum=0x0B55	55 0b
Message attribute, attPriority Length=2 bytes	01 0d 80 04 00 02 00 00 00
Data=0x0002 (Normal Priority)	02 00
Checksum=0x0002	02 00
Message attribute, attDateSent Length=14 bytes	01 05 80 03 00 0e 00 00 00
Data= Wednesday, 01/16/2008, 23:28:08	d8 07 01 00 10 00 17 00 1c 00 08 00 03 00
Checksum=0x012E	2e 01
Message attribute, attDateModified Length=14 bytes	01 20 80 03 00 0e 00 00 00
Data= Wednesday, 01/16/2008, 23:28:08	d8 07 01 00 10 00 17 00 1c 00 08 00 03 00
Checksum=0x012E	2e 01
Message attribute, Encapsulation, attMsgProps Length=136 bytes Number of properties=2	01 03 90 06 00 88 00 00 00 02 00 00 00
Message property, PidTagTnefCorrelationKey Number of property values=1 Size of first property=12	02 01 7f 00 01 00 00 00 0c 00 00 00
	38 71 6b 6a 30 30 73 67 6d 34 66 00
Message property, PidTagRtfCompressed Number of property values=1 Size of first property=93	02 01 09 10 01 00 00 00 5d 00 00 00
	59 00 00 00 b3 00 00 00 4c 5a 46 75 a9 be bb ed 87 00 0a 01 0d 03 43 74 65 78 74 01 f7 ff 02 a4 03 e4 05 eb 02 83 00 50 02 f3 06 b4 02 83 26 32 03 c5 02 00 63 68 0a c0 73 65 d8 74 30 20 07 13 02 80 7d 0a 80 08 cf 3f 09 d9 02 80 0a 84 0b 37 12 c2 01 d0 20 46 10 59 49 00 7d 18 20
Pad=3 bytes	00 00 00
Checksum for **attMsgProps**	f7 21

4 Security Considerations

4.1 attRecipTable, attFrom

Extreme care is recommended in the use of these attributes, particularly by the TNEF Reader, to avoid address or identity spoofing. More discussion of this can be found in [MS-OXCMSG] and section 5.2 TNEF encapsulation vs. Outer Wrapper attributes in Appendix A of this document.

5 Other Compatibility Issues

The preceding sections addressed issues of compatibility between the original implementation of TNEF and the richer implementation which is now used. This section discusses some compatibility issues within the richer implementation.

5.1 attOemCodepage handling

Attribute **attOemCodepage** has been handled inconsistently between multiple implementations of TNEF. This can be particularly troublesome when forwarding messages with attached messages between systems with different default codepages. To minimize problems caused by this inconsistent behavior, the following is the recommended way of handling string encoding:

5.1.1 Encoding by TNEF Writer

The TNEF Writer SHOULD write all ANSI strings, both the attribute strings and the encapsulated message property strings, with a consistent codepage and put its value into **attOemCodepage**. This SHOULD be set to a valid Windows ANSI codepage. The codepage chosen MUST NOT allow embedded zero bytes other than a zero terminator.1

If **PidTagInternetCpid** is being stored in the encapsulated properties, it SHOULD be set to a value which maps to the same language group or language as the codepage choosen for attOemCodepage. This codepage also determines the charset used in MIME if this TNEF is transmitted as a MIME entity body.

Any string that would suffer data loss if written using the consistent codepage SHOULD NOT be written to an attribute. All strings that would suffer data loss using the consistent codepage SHOULD be written to encapsulated properties with UNICODE data type.

For compatibility purposes, **attMessageClass** MUST always be written using the consistent codepage.

5.1.2 Decoding by TNEF Reader

For ANSI string decoding from the TNEF structure, the TNEF Reader MUST use the Windows codepage determined from the best of the following sources, listed in decreasing order of trustworthiness:

1. Any non US-ASCII MIME charset, if structure was transmitted as a MIME body part and charset name is available in the MIME, mapped to the most appropriate Windows codepage based on the criteria above;
2. **attOemCodepage**, if available and nonzero,
3. **PidTagInternetCpid**, if available in the encapsulated properties, mapped to the most appropriate Windows codepage based on the criteria above;
4. The most appropriate default codepage for the processing environment

5.2 TNEF Encapsulation vs. Outer Wrapper Attributes

It is possible to encapsulate properties in the TNEF structure representing data that is already conveyed as part of an outer wrapper such as MIME. Because in many cases the outer wrapper properties are subject to substantial validation during transport (reverse DNS lookups and other validation), they SHOULD be considered more trustworthy than information inside an attached file that otherwise could overwrite good information with bad.

As a general rule, TNEF Writers SHOULD NOT duplicate any data that is already being reliably transported outside of the structure and TNEF Readers SHOULD NOT override data from outside the structure with data from inside the structure. The individual implementations will determine how this rule is applied. For a detailed description of how this is handled in Outlook and Exchange for MIME messages, see [MS-OXCMAIL]. Some highlights of this behavior follow:

5.2.1 attBody Handling

When TNEF is being sent as a MIME body part, the **attBody** attribute is not written to TNEF or read from TNEF either by the client or server. The plaintext body is instead transmitted in a MIME body part. When TNEF is being used to encode an attached message, **attBody** MAY be written and read. See [MS-OXCMAIL] for a more detailed description.

5.2.2 attRecipTable Handling

When TNEF is being sent as a MIME body part, the **attRecipTable** attribute is not written to TNEF or read from TNEF either by the client or server, except when sending or receiving a legacy [non]delivery report message (**attMessageClass** beginning with “REPORT.” and ending in “.DR” or “.NDR”). The requisite message recipient information is transmitted in the MIME structure instead.

When TNEF is being sent as a MIME body part, when sending or receiving a legacy [non]delivery report message (**attMessageClass** beginning with “REPORT.” and ending in “.DR” or “.NDR”), recipient information from **attRecipTable** is used to populate the received message recipient table.

When TNEF is being used to encode an attached message, recipient information from **attRecipTable** is used to populate the received message recipient table.

5.2.3 attFrom Handling

The **attFrom** attribute is not written to TNEF or read from TNEF either by the client or server for MIME messages. The requisite message recipient information is transmitted in the MIME structure instead. When TNEF is being used to encode an attached message, the sender information is carried in encapsulated message properties in **attMsgProps**.

5.2.4 attSubject Handling

The **attSubject** attribute is not written to TNEF by the client or server for MIME messages, as the requisite subject information is transmitted in the MIME structure. When TNEF is being used to encode an attached message, the message subject information is carried in encapsulated message properties in **attMsgProps**. When the TNEF structure is being decoded, and **attSubject** is present, it will be used for the message if there is no MIME structure, if a MIME subject is not available, or if an attached message is being decoded.

6 Appendix A: Office/Exchange Behavior

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

- Office 2003 with Service Pack 3 applied
- Exchange 2003 with Service Pack 2 applied
- Office 2007 with Service Pack 1 applied
- Exchange 2007 with Service Pack 1 applied

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies 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.

1As of this writing, Exchange 2003 SP2 and Exchange 2007 SP1 recognizes any one of the following values:

Code	*Description*	*Code*	*Description*
874	Thai	1253	Greek
932	Japanese Shift-JIS	1254	Turkish
936	Simplified Chinese	1255	Hebrew
949	Korean	1256	Arabic
950	Traditional Chinese	1257	Baltic
1251	Cyrillic	1258	Vietnamese
1252	Latin-1 (similar to ISO 8859-1)		

6.1 Exchange 2003 Handling of ANSI Characters

Exchange 2003 implementation of TNEF would use the MIME charset to determine its mapping of strings in the attributes and of ANSI strings in the encapsulated properties, regardless of the value of **attOemCodePage**. This can cause some confusion, but by following the rules in Section 5 above for reading and writing TNEF, other implementations should not have any significant interoperability issues.
As of this writing, Exchange 2007 SP1 recognizes any one of the following values:

Code	*Description*	*Code*	*Description*
874	Thai	1253	Greek
932	Japanese Shift-JIS	1254	Turkish
936	Simplified Chinese	1255	Hebrew
949	Korean	1256	Arabic
950	Traditional Chinese	1257	Baltic
1251	Cyrillic	1258	Vietnamese
1252	Latin-1 (similar to ISO 8859-1)		

7 Index