[MS-OXORMMS]:
Rights-Managed E-Mail Object Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Updated references.
12/03/2008	1.03		Updated IP notice.
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	5.0.0	Major	Updated and revised the technical content.
05/05/2010	6.0.0	Major	Updated and revised the technical content.
08/04/2010	6.1	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction

This document specifies the Rights-Managed E-mail object protocol that is used by the client to create and consume a **rights-managed e-mail message**. A rights-managed **message** is used to protect e-mail content from inappropriate access, use, and distribution.

1.1 Glossary

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

attachment
Attachment object
binary large object (BLOB)
code page
flags
GUID
handle
header
Hypertext Markup Language (HTML)
little-endian
message
message body
Message object
metafile
named property
non-Unicode
offline
permissions
plain text
plain text message body
property (1)
property ID
publishing license (PL)
recipient
Rich Text Format (RTF)
rights-managed e-mail message
rights policy template
storage
store
stream (2)
Unicode
Use License (UL)
XML

The following terms are specific to this document:

Mapping mode: The way in which logical (device-independent) coordinates are mapped to device-specific coordinates.

storage container: See **storage**.

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

1.2 References

1.2.1 Normative References

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

[MS-CFB] Microsoft Corporation, "Compound File Binary File Format", June 2008, http://msdn.microsoft.com/en-us/library/dd942138(PROT.10).aspx

[MS-DTYP] Microsoft Corporation, "Windows Data Types", March 2007, http://msdn.microsoft.com/en-us/library/cc230273.aspx

[MS-OFFCRYPTO] Microsoft Corporation, "Office Document Cryptography Structure Specification", April 2008, http://msdn.microsoft.com/en-us/library/cc313071.aspx

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

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

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

[MS-OXMSG] Microsoft Corporation, ".MSG File Format", April 2008.

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

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

[MS-RMPR] Microsoft Corporation, "Rights Management Services (RMS): Client-to-Server Protocol Specification", March 2007, http://msdn.microsoft.com/en-us/library/cc209574.aspx

[MS-WMF] Microsoft Corporation, "Windows Metafile Format Specification", June 2007, http://msdn.microsoft.com/en-us/library/cc215212.aspx

[RFC1950] Deutsch, P., and Gailly, J-L., "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996, http://www.ietf.org/rfc/rfc1950.txt

[RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996, http://www.ietf.org/rfc/rfc1951.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

1.2.2 Informative References

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

[MSDN-DVASP] Microsoft Corporation, "DVASPECT Enumeration", http://msdn.microsoft.com/en-us/library/ms690318.aspx

1.3 Overview

This protocol enables the client to create and consume rights-managed e-mail messages.

When a client creates a rights-managed e-mail message, it encrypts and compresses the contents of the message (body, attachments, and so on) and stores the encrypted, compressed contents as part of the message that is sent to the **recipient(s)**. The client sets certain **properties** on the message to identify it as rights-managed.

When a client receives a rights-managed e-mail message, it decompresses and decrypts the encrypted **BLOB** and displays the content to the end user if the end user has sufficient permissions for the same. In addition, the client disables certain functionality on the rights-managed e-mail message to prevent the recipient from using the message in an unauthorized manner.

1.4 Relationship to Other Protocols

The Rights-Managed E-mail Object protocol specification relies on the following:

- An understanding of the **Message object** (as specified in [MS-OXCMSG]) so that the client can obtain a **handle** to the Message object and perform property operations on it.
- An understanding of **attachments** (as specified in [MS-OXCMSG]) and message properties (as specified in [MS-OXCMSG] and [MS-OXOMSG]) so that the client can handle attachments and perform property operations on **Attachment objects**.
- An understanding of the Rights Management Services (RMS) Client-Server protocol (as specified in [MS-RMPR]).
- An understanding of the compound file format (as specified in [MS-CFB]).

1.5 Prerequisites/Preconditions

This protocol specification assumes that the messaging client has previously logged on to the messaging server and has acquired a handle to the rights-managed e-mail message, as specified in [MS-OXCMSG] section 3.1.4.1.

This protocol relies on the Rights Management Services (RMS) client-server protocol (as specified in [MS-RMPR]) to create and consume rights-managed e-mail messages, and therefore assumes that the prerequisites of the RMS client-server protocol are met.

1.6 Applicability Statement

A client can use this protocol to create and consume rights-managed e-mail messages.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The properties specified in this protocol are transported between client and server, as specified in [MS-OXCMSG] section 2.1.

2.2 Message Syntax

A rights-managed e-mail message extends the Message and Attachment Object protocol specified in [MS-OXCMSG]. This protocol references commonly used data types, as specified in [MS-OXCDATA] section 2.11.1.

Unless otherwise specified, rights-managed e-mail message objects adhere to all property constraints specified in [MS-OXPROPS] and all property constraints specified in [MS-OXCMSG]. A rights-managed e-mail message object can also contain other properties, which are defined in [MS-OXPROPS], but these properties have no impact on the Rights-Managed E-mail Object protocol.<1>

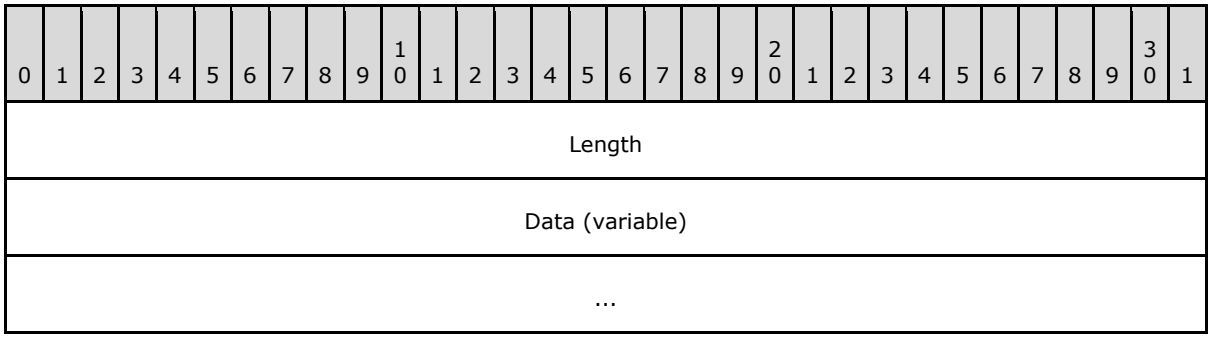
2.2.1 Rights-Managed E-Mail Message Property

The following property is specific to the Rights-Managed E-mail Object protocol.

2.2.1.1 PidNameRightsManagementLicense

Type: **PtypMultipleBinary**

This **named property** is used to cache the **Use License** for the rights-managed e-mail message. If the Use License is successfully obtained, this property SHOULD<2> be present on a rights-managed e-mail message object. If the property is present, the first value of this multiple binary property MUST contain the compressed (as specified in [RFC1950]) Use License for the rights-managed e-mail message. When uncompressed, the resulting data is a length-prefixed **Unicode** string that is formatted as follows.



Length (4 bytes): An unsigned integer that specifies the size of the **Data** field in **WCHAR**s.

Data (variable): A Unicode string containing the uncompressed Use License data.

2.2.2 Additional Property Constraints

This document specifies additional constraints on the following properties beyond what is specified in [MS-OXCMSG] section 2.2.1.

2.2.2.1 PidNameContentClass

Type: **PtypString**

The value of this property for a rights-managed e-mail message MUST be set to "rpmsg.message".

2.2.3 Attachment Object

A rights-managed e-mail message consists of a wrapper e-mail message with the original e-mail contents encrypted and compressed in an attachment. The attachment that contains the encrypted, compressed contents of the original email SHOULD be the only attachment on the wrapper e-mail message. If there are multiple attachments on the wrapper e-mail message, then the attachment that contains the encrypted, compressed contents of the original email MUST be the first attachment. This attachment has specific property values, as specified in the following sections, that distinguish it from the other attachments. For details about encryption, compression, decompression, and decryption of the original contents in the attachment, see section 3.1.4.

2.2.3.1 PidTagAttachLongFilename

Type: **PtypString**

The value of this property for a rights-managed e-mail message MUST be set to "message.rpmsg".

2.2.3.2 PidTagAttachMimeTag

Type: **PtypString**

The value of this property for a rights-managed e-mail message MUST be set to "application/x-microsoft-rpmsg-message".

2.2.4 Data Formats

This protocol references the following data formats, in addition to several commonly used data types that are defined in [MS-DTYP].

2.2.4.1 LPString

The LPString format represents a string that contains a 1-byte positive integer (**LengthOfString**) indicating the length of the string, followed by **LengthOfString** Unicode characters. This string is not null-terminated. The length of this string cannot be more than 255 characters.

2.2.4.2 non-Unicode LPString

The **non-Unicode** LPString format represents a string that contains a 1-byte positive integer (**LengthOfString**), indicating the length of the string, followed by **LengthOfString** non-Unicode characters. This string is not null-terminated. The length of this string cannot be more than 255 characters.

2.2.4.3 pipe-delimited string

The pipe-delimited string format represents a Unicode string containing multiple substrings, delimited by the pipe ("|") character. Each substring cannot contain the pipe character. This string always ends with the pipe character, even if there is only one substring. This is a length-prefixed string with the first byte containing the length of the Unicode characters. The length of this string cannot be more than 255 characters.

3 Protocol Details

The role of the client is to create a rights-managed e-mail message (by setting properties to distinguish the message from a normal message) and to identify and consume a rights-managed e-mail message when it is received.

3.1 Client Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

When a higher layer triggers the creation of rights-managed e-mail message, the original message, along with its attachments that are to be rights-managed, is encrypted and packaged in a **storage container**. This storage container is then compressed and stored as an attachment to a wrapper message that is marked with the specific property, as specified in section 2.2.2, which results in a rights-managed e-mail message. The attachment is also specified with certain properties, as specified in section 2.2.3, that distinguish it from a regular attachment.

3.1.4.1 Creating a Rights-Managed E-Mail Message

This section specifies how the client creates a rights-managed e-mail message when requested by the higher layers.

3.1.4.1.1 Encryption and Compression of the Original Message

When the higher layer creates a rights-managed e-mail message, handshaking between the client and the RMS server takes place, resulting in the generation of required certificates by the RMS server for creation and consumption of rights-managed content. For more information about this process, see [MS-RMPR].

When the client obtains the certificates that are required to create the rights-managed content:

- A storage container that is based on a **storage** structure referred to as the "compound file" (as specified in [MS-CFB]) MUST be created with the format as specified in section 3.1.4.1.3.
- The following components of the original message MUST be encrypted before they are included in the container:
  - **Message body**: Depending on the body format of the message, the message body is contained in one of these properties: PidTagBody, PidTagBodyHtml, PidTagRtfCompressed, PidTagRtfInSync, or the combination of PidTagRtfCompressed and PidTagRtfInSync.
  - attachments: If any attachments are present in the original message, they MUST be encrypted.

- The **publishing license** MUST be obtained for the encrypted content (as specified in [MS-RMPR]) and packaged in the storage container.
- This storage container MUST then be compressed (as specified in [RFC1951]) to reduce its size. See section 3.1.4.1.2.1 for more details.

3.1.4.1.2 Creation of the Wrapper E-mail Message

A wrapper e-mail message (that is, a Message object as specified in [MS-OXCMSG]) for the rights-managed e-mail message is created with an attachment that is formatted as specified in section 3.1.4.1.2.1. The PidTagAttachLongFilename ([MS-OXPROPS] section 2.661) of the attachment is set to "message.rpmsg" and PidTagAttachMimeTag ([MS-OXPROPS] section 2.668) of the attachment is set to "application/x-microsoft-rpmsg-message". The message.rpmsg attachment SHOULD be the only attachment created on the wrapper e-mail message. If multiple attachments are created on the wrapper e-mail message, the message.rpmsg attachment MUST be the first attachment and all subsequent attachments will be ignored when the client opens a rights-managed e-mail message (as specified in section 3.1.4.2). The PidNameContentClass ([MS-OXPROPS] section 2.432) of the wrapper message is set to "rpmsg.message".

3.1.4.1.2.1 Format of the message.rpmsg Attachment

The message.rpmsg attachment contains a prefix that is formatted as follows:

CHAR STRING header - Value is "\x76\xE8\x04\x60\xC4\x11\xE3\x86" in **little-endian** format.

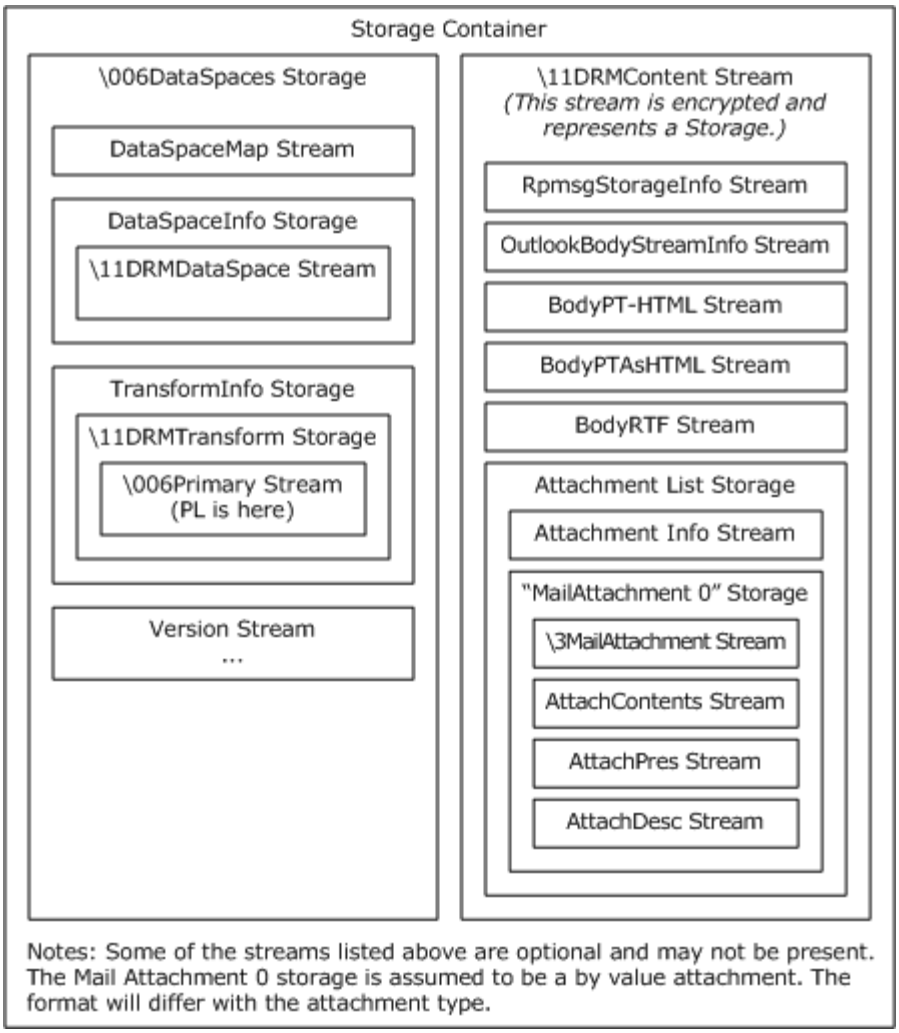
The prefix is followed by one or more blocks of data. The uncompressed content is divided into segments of 4096 bytes. Each block of the compressed **stream** contains the following:

- DWORD ULCheck - Value is 0x00000FA0.
- DWORD SizeAfterInflation - Size of the uncompressed data segment. Usually 4096; the last block can be less.
- DWORD SizeBeforeInflation - Size of the compressed data segment.
- compressed (as specified in [RFC1950]) data segment (that is, the compressed bits of the storage container that is specified in section 3.1.4.1.3).

The client compresses (as specified in [RFC1950] and [RFC1951]) the bits for each successive block.

3.1.4.1.3 Format of the Storage Container

Figure 1 shows the format of the storage container.



The components listed in the following table MUST be present in the uncompressed storage container.

Stream/Storage	Name	Description	Format
storage	"\006DataSpaces"	Contains data, such as the PL and transformation information for the document.	See [MS-OFFCRYPTO] for more details.
stream	"\11DRMContent"	Contains the encrypted message body and attachments.	See section 3.1.4.1.3.1 for details.

3.1.4.1.3.1 \11DRMContent Storage

The \11DRMContent storage contains the encrypted e-mail message body and attachments. Before encryption, the \11DRMContent has the components specified in the following table.

Name	Stream/Storage	Description
OutlookBodyStreamInfo	This stream MUST be present in the storage.	This stream contains two consecutive values: The first value is of type **WORD** and contains the message body format. If the body format is **plain text**, the value MUST be 0x0001. If the body format is **HTML**, the value MUST be 0x0002. If the body format is **RTF**, the value MUST be 0x0003. The second value is of type **DWORD** whose value MUST correspond to PidTagInternetCodepage, if present; otherwise it MUST be set to the active **code page** of the system.
BodyPT-HTML	This stream MUST be present in the storage.	The contents of this stream are based on the body format as specified in OutlookBodyStreamInfo stream. If the body format is plain text, this stream MUST contain the plain text version of the message body that is present in the PidTagBody property, as specified in [MS-OXBBODY] section 1.3. If the body format is HTML, this stream MUST contain the HTML version of the message body that is present in the PidTagHtml property, as specified in [MS-OXBBODY] section 1.3. If the body format is RTF, this stream MUST contain an HTML version of the RTF message body.<3>
BodyRtf	If the message body format specified in OutlookBodyStreamInfo is RTF, this stream MUST be present in the storage.	Contains the RTF representation of the message body that is present in the PidTagRtfCompressed property, as specified in [MS-OXBBODY] section 1.3.
BodyPTAsHTML	If the message body format specified in OutlookBodyStreamInfo is plain text, this stream MUST be present in the storage.	This stream contains an HTML version of a **plain text message body**. The client MUST ignore this stream on receipt.
RpmsgStorageInfo	This stream MUST be present in the storage.	This stream contains implementation specific details. It MUST contain the following byte stream: 1F 32 DE 15 02 00 00 00 02 00 00 00 00 00 00 00.
WordMailRightsIndex	This stream SHOULD<4> be present in the storage if the message is a Reply to a rights-managed e-mail message; otherwise, it MUST NOT be included.	When replying to a rights-managed e-mail message, the replier cannot copy or print the original message included within/below the reply. To differentiate between this protected and unprotected content in saved e-mail messages, WordMailRightsIndex contains first and last character position pairs that bind content within the message. The first

Name	Stream/Storage	Description
		pair represents the beginning and end character positions of the original message. The remaining character pairs represent the bounds of the inline comments in the original message. Multiple pairs exist when inline comments are used. The stream MUST contain the following: A **ULONG** to represent the number of pairs, followed by the character position pairs. Each pair consists of two character positions, each of type **ULONG**. The values are stored in the little-endian format.
Attachment List	This storage MUST be present if the message has any attachment.	Contains the attachment storage collection of the message. See section 3.1.4.1.3.2 for details.

3.1.4.1.3.2 Attachments to the Rights-Managed E-Mail Message

All attachments in the message MUST be stored in the "Attachment List" storage. The contents of the attachment MUST be encrypted with the same PL as the Message object if the associated application supports rights management.<5>

The components of the "Attachment List" storage are specified in the following table.

Name	Stream/Storage	Description
Attachment Info	This stream MUST be present.	See section 3.1.4.1.3.3 for details.
MailAttachment N	This storage MUST be present.	N represents the "attachment number" that starts from zero and is incremented with each attachment. See section 3.1.4.1.3.4 for details.

3.1.4.1.3.3 Attachment Info

This stream provides a table of contents for the Attachment List storage. This stream MUST contain the following in the order given:

- **ULONG** AttachmentCount, which gives the number of attachments. If this value is 0xFFFFFFFF, the message body format MUST be RTF. The number of attachments in case of RTF messages is in the **DWORD** NumberOfAttachments, as specified in the next table.
- Pipe-delimited string (as specified in section 2.2.4.3) containing the list of attachments in the form of "Mail Attachment N", where N represents the attachment number starting from zero. The format of the Unicode pipe-delimited string is as follows:

MailAttachment 0|MailAttachment 1|... MailAttachment N|

If the message body format specified in OutlookBodyStreamInfo is RTF, the following information is appended to the stream. All values are stored in the little-endian format.

Name	Format	Description
NumberOfAttachments	**DWORD**	This value contains the number of attachments in the RTF message.

The following are then repeated for each attachment that is present in the RTF message:

Name	Format	Description
CharacterPosition	**DWORD**	This is the location in the RTF stream in which the embedded object appears. This corresponds to the PidTagRenderingPosition, as specified in [MS-OXCMSG] section 2.2.2.16.
Objf	**DWORD**	This value represents the way the contents of an attachment can be accessed. The following are the possible values: A value of 0x00000001 MUST be matched to the attachment with PidTagAttachMethod afOle. A value of 0x00000004 MUST be matched to the attachment with PidTagAttachMethod afByValue. A value of 0x00000008 MUST be matched to the attachment with PidTagAttachMethod afEmbeddedMessage. Objf also has other client-specific **flags** logically ORed to the above values which are implementation-specific and can be ignored.
Aspect	**DWORD**	This contains the objects draw aspect. If the Objf value is 0x00000004 or 0x00000008, it is set to DVASPECT_ICON (as described in [MSDN-DVASP]). If the Objf value is 0x00000001, it is set to DVASPECT_CONTENT.
SizeAlongXAxis	**DWORD**	This value is the length of the **metafile** that is displayed in the message body. Metrics are based on the **Mapping mode** of the metafile.
SizeAlongYAxis	**DWORD**	This value is the height of the metafile that is displayed in the message body. Metrics are based on the Mapping mode of the metafile.

3.1.4.1.3.4 MailAttachment Structure

The structure of "MailAttachment N" storage is dependent on the way the contents of the attachment can be accessed. The different ways are specified by the PidTagAttachMethod property, as specified in [MS-OXCMSG] section 2.2.2.9. A rights-managed e-mail message MUST allow only the following values for the PidTagAttachMethod property:

- afByValue
- afEmbeddedMessage
- afOle

Treatment of each type of attachment is described separately in the following subsections.

3.1.4.1.3.4.1 afByValue

The following table specifies the components of the "MailAttachment N" storage for the attachment for which the PidTagAttachMethod property is afByValue.

Name	Stream/Storage	Description
\3MailAttachment	This stream MUST be present	Attachment **header** stream. See section 3.1.4.1.3.4.1.1.
AttachPres	This stream MUST be present	This stores the attachment's icon. See section 3.1.4.1.3.4.1.2.
AttachDesc	This stream MUST be present	Information about the attachment. See section 3.1.4.1.3.4.1.3.
AttachContents	This stream MUST be present	The actual contents of the attachment. See section 3.1.4.1.3.4.1.4.

Other streams can be present in the storage but they are client-specific.

3.1.4.1.3.4.1.1 \3MailAttachment Stream

The following table specifies the format of the components in the \3MailAttachment stream in the order in which they appear. The values are stored in little-endian format.

Component	Format	Comments
Attachment Number	**DWORD**	Represents the index of the attachment at the time of attaching. This can be set to 0x00000000. Client ignores this value on receipt.
Reference Flag	**DWORD**	This is an implementation-specific flag. This can be set to 0x00000000. Client ignores this value on receipt.

3.1.4.1.3.4.1.2 AttachPres

This stream stores the attachment icon for user presentation in the Windows metafile format, as specified in [MS-WMF].

3.1.4.1.3.4.1.3 AttachDesc

This stream stores information about the attachment. The following table specifies the format of the components of the AttachDesc stream in the order in which they appear.

Component	Format	Comments
Stream Version	**USHORT**	When creating a rights-managed e-mail message, this value MUST always be set to 0x0203. The value is stored in the little-endian format.
Long Path Name	non-Unicode LPString (section 2.2.4.2)	Long Path Name SHOULD<6> contain the value of the PidTagAttachLongPathname property of the attachment, if present; otherwise, it MUST be 0x00.
Path Name	non-Unicode LPString	Path Name MUST contain the value of the PidTagAttachPathname property of the attachment, if present; otherwise, it MUST be 0x00.
Display name	non-Unicode LPString	Display name MUST contain the value of the PidTagDisplayName property of the attachment, if present; otherwise, it MUST be 0x00.

Component	Format	Comments
Long File Name	non-Unicode LPString	Long File Name MUST contain the value of the PidTagAttachLongFilename property of the attachment, if present; otherwise, it MUST be 0x00.
File Name	non-Unicode LPString	File Name MUST contain the value of the PidTagAttachFilename property of the attachment, if present; otherwise, it MUST be 0x00.
Extension	non-Unicode LPString	Extension MUST contain the value of the PidTagAttachExtension property of the attachment, if present; otherwise, it MUST be 0x00.
File Creation Time	64-bit value	File Creation Time MUST contain the value of the PidTagCreationTime property of the attachment, if present; otherwise, it MUST be 0x0000000000000000. This is stored in little-endian format.
File Last Modified Time	64-bit value	File Last Modified Time MUST contain the value of the PidTagLastModificationTime property of the attachment, if present; otherwise, it MUST be 0x0000000000000000. This is stored in little-endian format.
Attach Method	**ULONG**	Attach Method MUST contain the value of the PidTagAttachMethod property of the attachment stored in little-endian format.
Content ID	LPString (section 2.2.4.1)	Content ID MUST contain the PidTagAttachContentId property value, if present; otherwise it MUST be 0x00.
Content Location	LPString	Content Location MUST contain the PidTagAttachContentLocation property value, if present; otherwise, it MUST be 0x00.
Long Path Name	LPString	Long Path Name SHOULD<7> contain the value of the PidTagAttachLongPathname property of the attachment, if present; otherwise, it MUST be 0x00.
Path name	LPString	Path Name MUST contain the value of the PidTagAttachPathname property of the attachment, if present; otherwise, it MUST be 0x00.
display name	LPString	Display name MUST contain the value of the PidTagDisplayName property of the attachment, if present; otherwise, it MUST be 0x00.
Long File name	LPString	Long File name MUST contain the value of the PidTagAttachLongFilename property of the attachment, if present; otherwise, it MUST be 0x00.
File name	LPString	File name MUST contain the value of the PidTagAttachFilename property of the attachment, if present; otherwise, it MUST be 0x00.
Extension	LPString	Extension MUST contain the value of the PidTagAttachExtension property of the attachment, if present; otherwise, it MUST be 0x00.
Image Preview Small	LPString	This value MUST contain the file name that contains the small Image Preview of the attachment, if present; otherwise, it MUST be 0x00.
Image Preview Medium	LPString	This value MUST contain the file name of the medium Image preview of the attachment, if present; otherwise, it MUST be 0x00.
Image Preview Large	LPString	This value MUST contain the file name of the large Image preview of the attachment, if present; otherwise, it MUST be 0x00.

Component	Format	Comments
Rendered	**LONG**	This value MUST be 0x00000001 if the attachment is rendered inline (only valid for HTML images). Otherwise, it MUST be set to 0x00000000. This value is stored in little-endian format.
flags	**LONG**	This field contains certain implementation-specific flags that correspond to the attachment. When creating a rights-managed e-mail message, this value can be set to 0x00000000. This value is stored in little-endian format.

3.1.4.1.3.4.1.4 AttachContents

This stream stores the actual bits of the attachment, as specified in the following table.

Component	Format	Comments
attachment	Binary data	The attachment contents are stored here. This is the same as the PidTagAttachDataBinary property value.

3.1.4.1.3.4.2 afEmbeddedMessage

The "MailAttachment N" storage structure for attachments with PidTagAttachMethod property afEmbeddedMessage is the same as that for the attachment with PidTagAttachMethod as afByValue, with the following exceptions:

- In the AttachDesc stream, the Attach Method field MUST be set to "0x0005" to represent that the attachment is an embedded message.
- The AttachContents stream MUST be replaced by an .msg storage file (the structure of which is specified in [MS-OXMSG]), which is the embedded message converted into storage.
- On receipt of the rights-managed e-mail message with .msg attachments, the client creates an embedded message Attachment object from the contents of the .msg storage file.

3.1.4.1.3.4.3 afOle

This applies to RTF message body formats only.

If the attachment's PidTagAttachTag property value is afStorage ([MS-OXCMSG] section 2.2.2.15), then "MailAttachment N" storage MUST contain the value of PidTagAttachDataBinary ([MS-OXCMSG] section 2.2.2.7) converted to a compound file storage ([MS-CFB]).

If the PidTagAttachTag property value is not afStorage, then "MailAttachment N" storage MUST contain a copy of PidTagAttachDataObject, as specified in [MS-OXCMSG] section 2.2.2.8.

3.1.4.2 Opening a Rights-Managed E-Mail Message

This section specifies how a rights-managed e-mail message is consumed when a user or user agent invokes an event to open a rights-managed e-mail message.

When an event to open a message is triggered, the client scans the message for the properties specified in section 2.2.1 and 2.2.2 and scans the first attachment for the properties specified in section 2.2.3. The client uses the values of these properties to determine whether a message is

rights-managed. Following this determination, the client proceeds to open the message, as specified in section 3.1.4.2.1.

3.1.4.2.1 Decompression and Decryption of the Message

After the message has been identified as a rights-managed e-mail message:

- To get the storage container (as specified in section 3.1.4.1.3), the client MUST process the "message.rpmsg" attachment (as specified in section 3.1.4.1.2.1), decompressing the compressed bits (as specified in [RFC1950] and [RFC1951]) in each successive data block.
- If the PidNameRightsManagementLicense property is present, this indicates that the Use License (UL) has already been obtained. If it is not present, the client MUST obtain the required UL from the RMS server by using the publishing license (PL) in the container, as specified in [MS-RMPR] section 3.4.4.1.
- The client SHOULD<8> then cache the UL in the PidNameRightsManagementLicense property, as specified in section 2.2.1.1, if the UL is not already present. A user can open the message **offline** if the UL is cached. When the message is opened, the client SHOULD store the UL in the RMS License **store** so that the license can be used to open any rights-managed attachments in the message.
- By using the UL, the messaging client decrypts the encrypted content as specified in [MS-RMPR].

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

The RMS server is responsible for issuing the various certificates and licenses required for the creation and consumption of rights-managed e-mail messages. The role and details of the RMS server are specified in detail in [MS-RMPR].

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Creating a Rights-Managed E-Mail Message

Joe creates a rights-managed e-mail message and saves it. The following is a description of what a protocol client might do to accomplish Joe's intentions, followed by the responses a protocol server might return.

Before manipulating rights-managed Message objects, the protocol client needs a request for the protocol server to perform a mapping from named properties to **property IDs**, by using RopGetPropertyIDsFromNames.

Property	Property Set GUID	Name or ID
PidNameContentClass	{00020386-0000-0000-c000-000000000046}	Content class

The protocol server might return the following property IDs in response to RopGetPropertyIDsFromNames. The actual IDs are completely at the discretion of the protocol server.

Property	Property ID
PidNameContentClass	0x806C

To create a rights-managed object, the protocol client uses RopCreateMessage. The protocol server returns a success code and a handle to the Message object. The protocol client then uses RopSetProperties to transmit the data to the protocol server.

Property	Property ID	Type	Value
PidNameContentClass	0x806C	0x001F	rpmsg.message

In order to create message.rpmsg attachment, the protocol client uses RopCreateAttachment to create the Attachment object. Then the protocol client uses RopOpenStream and RopSetStreamSize followed by RopWriteStream to write out the contents into the attachment. The protocol client also asks the protocol server for specific attachment properties, which are then set by using RopSetProperties.

Property	Property ID	Type	Value
PidTagAttachMimeTag	0x370E	0x001F	application/x-microsoft-rpmsg-message
PidTagAttachLongFilename	0x3703	0x001F	message.rpmsg

The protocol client uses RopSaveChangesAttachment to save the Attachment object.

When Joe is ready to save his changes, the protocol client uses RopSaveChangesMessage to commit the properties on the protocol server, and then uses RopRelease to release the object.

The values of some properties will change during the execution of RopSaveChangesMessage, but none of the properties that are specified in this document will change.

5 Security

5.1 Security Considerations for Implementers

The key used to encrypt the content, which is generated by the RMS client, has to be different every time a rights-managed e-mail message is created and whenever any component of the **rights policy template** changes. Security considerations of the RMS client and server also figure in this protocol and are specified in [MS-RMPR].

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products:

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

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

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

<1> Section 2.2: Office Outlook 2007 and Outlook 2010 set the following properties (in addition to those specified in sections 2.2.1.1 and 2.2.2.1) on a new rights-managed e-mail message, regardless of user input: PidLidAgingDontAgeMe, PidLidCurrentVersion, PidLidCurrentVersionName, PidLidPrivate, PidLidSideEffects, PidTagAlternateRecipientAllowed, PidTagClientSubmitTime, PidTagDeleteAfterSubmit, PidTagImportance, PidTagMessageDeliveryTime, PidTagPriority, PidTagReadReceiptRequested, PidTagSensitivity, PidLidReminderDelta, PidLidReminderSet, PidLidTaskMode.

<2> Section 2.2.1.1: Office Outlook 2003 does not use this property to cache the Use License.

<3> Section 3.1.4.1.3.1: Outlook Web Access does not recognize the rights-managed e-mail message. To read the rights-managed e-mail message in Outlook Web Access, a Rights Managed add-on for Internet Explorer has to be installed. This stream is meant for the Rights Managed add-on for Internet Explorer.

<4> Section 3.1.4.1.3.1: This stream is not included in Office Outlook 2003 with a reply to a rights-managed e-mail message.

<5> Section 3.1.4.1.3.2: In the typical case, the unmodified bits of the attachment are stored in this stream. However, if the file type of the attachment supports rights management, the attachment is also rights-managed. Examples of such file types are Microsoft Word, Microsoft Excel, and Microsoft PowerPoint binary Office 97-2003 file format; Office 2007 new Open **XML** files formats (for example, .docx); Microsoft InfoPath 2007 forms and templates; and XPS documents. When rights- managed protection is applied to attachments, the same issuance license that is used to protect the message itself is used. The structure of each encrypted attachment conforms to the specifications for its file type. To view a rights-managed attachment, the file has to be opened and unencrypted in its native viewer.

<6> Section 3.1.4.1.3.4.1.3: In some cases, Office Outlook 2003, Office Outlook 2007, and Outlook 2010 write the full directory path to the attachment on the local computer.

<7> Section 3.1.4.1.3.4.1.3: In some cases, Office Outlook 2003, Office Outlook 2007, and Outlook 2010 write the full directory path to the attachment on the local computer.

<8> Section 3.1.4.2.1: Office Outlook 2003 does not use this property to cache the UL.

7 Change Tracking

This section identifies changes that were made to the [MS-OXORMMS] protocol document between the May 2010 and August 2010 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

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

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

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

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

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type "Editorially updated."

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

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.1 Glossary	57230 Added "attachment", "flags", "handle", "header", "non-Unicode", "offline", "permissions", "plain text message body", "recipient", "store", and "XML" to the list of terms that are defined in [MS-OXGLOS].	N	Content update.
1.2.1 Normative References	55751 Moved [MS-OXGLOS] from Normative References section to Informative References section.	N	Content update.
1.2.1 Normative References	48653 Added the reference [MS-OXCDATA].	N	Content update.
1.5 Prerequisites/Preconditions	48284 Added specific section reference for [MS-OXCMSG].	N	Content update.
2.1 Transport	48284 Added specific section reference for [MS-OXCMSG].	N	Content update.
2.2 Message Syntax	48653 Added reference to [MS-OXCDATA].	N	Content update.
2.2 Message Syntax	48263 Updated section references in behavior note.	N	Product behavior note

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
			updated.
2.2.2 Additional Property Constraints	48284 Added specific section reference for [MS-OXCMSG].	N	Content update.
2.2.2.1 PidNameContentClass	48653 Added data type information.	N	Content update.
2.2.3.1 PidTagAttachLongFilename	48653 Added data type information.	N	Content update.
2.2.3.2 PidTagAttachMimeTag	48653 Added data type information.	N	Content update.
3.1.1 Abstract Data Model	48265 Moved content to Higher-Layer Triggered Events section.	N	Content removed.
3.1.4 Higher-Layer Triggered Events	48265 Added content that was moved from the Abstract Data Model section.	N	Content update.
3.1.4.1.1 Encryption and Compression of the Original Message	51205 Added normative language.	N	Content update.
3.1.4.1.3.1 \11DRMContent Storage	48284 Added specific section references for [MS-OXBBODY].	N	Content update.
3.1.4.1.3.2 Attachments to the Rights-Managed E-Mail Message	48276 Changed "can" to "MUST".	N	Content update.
3.1.4.1.3.3 Attachment Info	48284 Added specific section reference for [MS-OXCMSG].	N	Content update.
3.1.4.1.3.4 MailAttachment Structure	48284 Added specific section reference for [MS-OXCMSG].	N	Content update.
3.1.4.2.1 Decompression and Decryption of the Message	51205 Added normative language.	N	Content update.
3.1.1.1 Managing a Rights-Managed E-Mail Message	48265 Moved contents to Higher-Layer Triggered Events section.	N	Content removed.

8 Index