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

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

Table of 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 object
Binary Large Object (BLOB)
code page
GUID
Hypertext Markup Language (HTML)
little-endian
message
message body
Message object
metafile
named property
plain text
property
property ID
publishing license (PL)
Rich Text Format (RTF)
rights policy template
Unicode
Use License (UL)

The following data types are defined in [MS-DTYP]:

BYTE
DWORD
LONG
ULONG
USHORT

The following terms are specific to this document:

LPString: 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.

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

non-Unicode LPString: 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.

pipe-delimited string: A **Unicode** string containing multiple sub-strings, delimited by the pipe ("|") character. Each sub-string cannot contain the pipe character. This string always ends with the pipe character, even if there is only one sub-string. 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.

rights-managed e-mail message: An e-mail **message** that specifies permissions that are designed to protect its content from inappropriate access, use, and distribution.

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-OFFCRYPTO] Microsoft Corporation, "Office Document Cryptography Structure Specification", April 2008, http://go.microsoft.com/fwlink/?LinkId=115029.

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

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

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

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

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

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

[MS-RMPR] Microsoft Corporation, "Rights Management Services (RMS): Client-to-Server Protocol Specification", March 2007, http://go.microsoft.com/fwlink/?LinkId=111755.

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

[MSFT-CFB] Microsoft Corporation, "Compound File Binary File Format", February 2008, http://go.microsoft.com/fwlink/?LinkId=111739.

[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.

[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.

[XRML] ContentGuard, Inc., "XrML... eXtensible rights Markup Language", 2005, http://www.xrml.org/XrML_12.asp.

1.2.2 Informative References

[MSDN-DVASP] Microsoft Corporation, "DVASPECT", http://go.microsoft.com/fwlink/?LinkId=111748.

[MSDN-OLE] Microsoft Corporation, "OleConvertIStorageToOLESTREAM Function", http://go.microsoft.com/fwlink/?LinkId=111754.

1.3 Protocol 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 the contents of the **message** (body, attachments, and so on) and stores the encrypted 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 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 objects and performs **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 [MSFT-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]).

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].

2.2 Message Syntax

A **rights-managed e-mail message** extends the Message and Attachment Object protocol specified in [MS-OXCMSG].

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 MAY<1> also contain other properties, which are defined in [MS-OXPROPS], but these properties have no impact on the Rights-Managed E-mail Object protocol

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

A **PtypMultipleBinary named property**. This **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 ZLIB (as specified in [RFC1950]) compressed Use License for the rights-managed e-mail message.

2.2.2 Additional Property Constraints

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

2.2.2.1 PidNameContentClass

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 in an attachment. A rights-managed e-mail message, therefore, MUST have at least one attachment. This attachment has specific **property** values, as specified in the following sections, that distinguish it from the other attachments. For details about encryption and decryption of the original contents in the attachment, see section 3.1.4.

2.2.3.1 PidTagAttachLongFilename

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

2.2.3.2 PidTagAttachMimeTag

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

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

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

3.1.1.1 Managing a Rights-Managed E-Mail Message

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.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

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 of the 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 [MSFT-CFB]) is created with the format as specified in section 3.1.4.1.2.

- 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 there are any attachments 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 using ZLIB<3> to reduce its size.
- A wrapper e-mail **message** for the rights-managed e-mail message MUST be created with the compressed stream as an attachment with the **PidTagAttachLongFilename** set to "message.rpmsg" and **PidTagAttachMimeTag** set to "application/x-microsoft-rpmsg-message". The **PidNameContentClass** of the wrapper message is set to "rpmsg.message".

3.1.4.1.2 Format of the Storage Container

The "message.rpmsg" attachment is a ZLIB compressed file that contains the standard ZLIB header <4> followed by the compressed storage container. Figure 1 shows the format of the storage container.

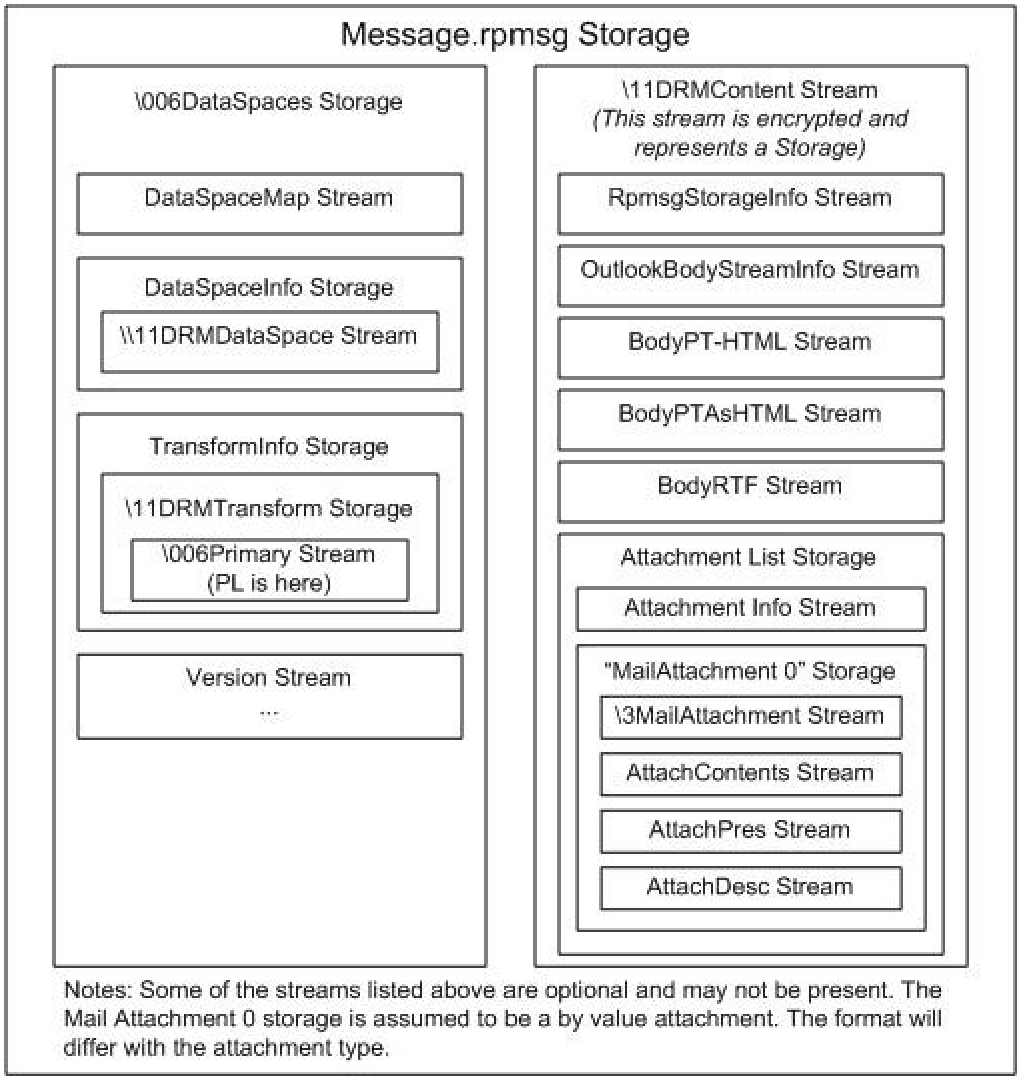

Figure 1: Format of the message.rpmsg container

The following components MUST be present in the uncompressed message.rpmsg storage:

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 3.1.4.1.4.1 for details.

3.1.4.1.2.1 \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]. 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]. If the body format is RTF, this stream MUST contain an HTML version of the RTF message body. <5>
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].
BodyPTAsHTML	If the message body format specified in	This stream contains an HTML version of a plain text message body. The client

	OutlookBodyStreamInfo is plain text, this stream MUST be present in the storage.	MUST ignore this stream on receipt. <6>
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 <7>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 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.

3.1.4.1.2.2 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 can be encrypted with the same **PL** as the **Message object** if the associated application supports rights management <8>.

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.

3.1.4.1.2.3 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** 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 MUST contain 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 MUST correspond to the **PidTagRenderingPosition**, as specified in [MS-OXCMSG].
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 MUST contain 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 MUST be 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 MUST be the height of the metafile that is displayed in the message body. Metrics are based on the mapping mode of the metafile.

3.1.4.1.2.4 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]. 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.

3.1.4.1.2.4.1 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 3.1.4.1.4.3.1.1.
AttachPres	This stream MUST be present	This stores the attachment's icon. See section 3.1.4.1.4.3.1.2.
AttachDesc	This stream MUST	Information about the attachment. See section

	be present	3.1.4.1.4.3.1.3.
AttachContents	This stream MUST be present	The actual contents of the attachment. See section 3.1.4.1.4.3.1.4.

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

3.1.4.1.2.4.1.1 \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 MAY be set to 0x00000000. Client ignores this value on receipt.
Reference Flag	**DWORD**	This is an implementation-specific flag. This MAY be set to 0x00000000. Client ignores this value on receipt.

3.1.4.1.2.4.1.2 AttachPres

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

3.1.4.1.2.4.1.3 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**	Long Path Name SHOULD <9>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.
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	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<10>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.
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 MAY be set to 0x00000000. This value is stored in little-endian format.

3.1.4.1.2.4.1.4 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 MUST be stored here. This is the same as the **PidTagAttachDataBinary property** value.

3.1.4.1.2.4.2 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.

3.1.4.1.2.4.3 afOle

This applies to **RTF message body** formats only.

If the attachments **PidTagAttachTag property** value is OLESTORAGE (as specified in [MS-OXCMSG]), then "MailAttachment N" storage MUST contain the value of **PidTagAttachDataBinary** (as specified in [MS-OXCMSG]) converted to a compound file storage. For more information, see [MSDN-OLE].

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

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 MUST scan the message for the properties specified in section 2.2.1 and 2.2.2 and MUST scan the attachment for the properties specified in section 2.2.3. The client MUST use the values of these properties to determine whether a message is rights-managed. Following this determination, the client MUST proceed to open the message, as specified in section 3.1.4.2.1.

3.1.4.2.1 Decryption of the Message

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

- The client MUST ZLIB decompress the "message.rpmsg" attachment to get the storage container, as specified in section 3.1.4.1.4.

- 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 section 3.1.7.9 of [MS-RMPR].

- The client SHOULD<11> then cache the UL in the **PidNameRightsManagementLicense** property, as specified in section 2.2.2.2, if the UL is not already present. A user can open the message offline if the UL is cached <12>.

- By using the UL, the messaging client MUST decrypt 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** identifiers, by using **RopGetPropertyIDsOfNames**.

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-OXRMPR].

5.2 Index of Security Parameters

None.

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.

<1> Section 2.2: Microsoft Office Outlook 2007 SP1 has the following properties in addition to those mentioned in sections 2.2.2.1, 2.2.2.2 and 2.2.2.3 that it will set 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**, **PidLidReminderNextTime**, **PidLidTaskMode**.

<2> Section 2.2.1.1: Outlook 2007 SP1 does not use this property to cache the **Use License**.

<3>Section 3.1.4.1.1: ZLIB library version that is greater than 1.2.3 has to be used.

<4> Section 3.1.4.1.2: The specific format of the message.rpmsg header is described as follows:

CHAR STRING Header - Value is "\x76\xE8\x04\x60\xC4\x11\xE3\x86" in **little-endian** format.
DWORD ULCheck - Value is 0x00000FA0
DWORD SizeBeforeInflation - Size of byte stream after decompressing.
DWORD SizeAfterInflation - Size of byte steam after compressing.

The bytes following the header contain the compressed bits of the storage container. The size of the compressed bits is determined by SizeBeforeInflation.

<5> Section 3.1.4.1.2.1: 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.

<6> Section 3.1.4.1.2.1: This stream is meant only for the Rights Managed add-on for Internet Explorer.

<7> Section 3.1.4.1.2.1: This stream is not included Microsoft Office Outlook 2003 SP2 with a reply to a **rights-managed e-mail message**.

<8> Section 3.1.4.1.2.2: 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; Microsoft 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.

<9>Section 3.1.4.1.2.4.1.3: In some cases, Outlook 2003 SP2 and Outlook 2007 SP1 write the full directory path to the attachment on the local computer.

<10> Section 3.1.4.1.2.4.1.3: In some cases, Outlook 2003 SP2 and Outlook 2007 SP1 write the full directory path to the attachment on the local computer.

<11>Section 3.1.4.2.1: Outlook 2007 SP1 does not use this property to cache the **Use License (UL)**.

<12> Section 3.1.4.2.1: Outlook 2003 SP2 and Outlook 2007 SP1 recognize the incoming e-mail **message** as a **rights-managed e-mail message** and tries to pre-license the e-mail message even before the message is opened. The **Use License (UL)** is then cached in the **PidNameRightsManagementLicense** property. When the e-mail message is opened, Outlook also stores this license in the RMS License store so that the license can be used to open any rights-managed attachments in the message.

Index