[MS-OXBBODY]:
Best Body Retrieval Algorithm

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 Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Revised and edited technical content.
10/01/2008	1.03		Revised and edited technical content.
12/03/2008	1.04		Updated IP notice.
03/04/2009	1.05		Revised and edited technical content.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0.1	Minor	Revised and edited for technical content.
11/04/2009	3.1.1	Minor	Updated the technical content.
02/10/2010	3.1.1	None	Version 3.1.1 release
05/05/2010	3.1.2	Editorial	Revised and edited the technical content.
08/04/2010	3.2	Minor	Clarified the meaning of the technical content.
11/03/2010	3.2	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	3.3	Minor	Clarified the meaning of the technical content.
08/05/2011	4.0	Major	Significantly changed the technical content.
10/07/2011	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	5.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Best Body Retrieval Algorithm determines the best format of a **message body (2)**. This algorithm enables clients to determine the format of the message body (2) that is most like the original message, and maintains the richness of the text and formatting in the original message.

Exactly how the server converts message text from one format to another, and to what extent formatting is preserved in the conversion, is implementation-dependent.

Section 2 of this specification is normative and contains RFC 2119 language. Section 1.6 is also normative but cannot contain RFC 2119 language. All other sections and examples in this specification are informative.

1.1 Glossary

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

best body
Hypertext Markup Language (HTML)
message body
Message object
plain text
recipient
remote operation (ROP)
Rich Text Format (RTF)
ROP request
S/MIME (Secure/Multipurpose Internet Mail Extensions)

The following terms are specific to this document:

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

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

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-OXCDATA] Microsoft Corporation, "Data Structures".

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

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

[MS-OXORMMS] Microsoft Corporation, "Rights-Managed E-Mail Object Protocol Specification".

[MS-OXOSMIME] Microsoft Corporation, "S/MIME E-Mail Object Algorithm".

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

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

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

1.2.2 Informative References

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

1.3 Overview

The **best body** algorithm determines the original, primary, or best format in which to transmit a message body (2) to a client. The best body algorithm enables clients that are capable of handling multiple message body (2) formats to determine which of the formats that they support is most like the original message. Requesting a message by using the best body algorithm maintains as much of the richness of the text and formatting in the original message as possible. The algorithm uses a combination of **remote operations (ROPs)** and property values to determine the best body format of the message. The best body format can be one of the following formats:

- **Plain text** – This format cannot display colors, different fonts, or emphasis such as bold or italic text. Plain text is the most accepted messaging format. Most e-mail message readers can display messages in plain text format.
- **Rich Text Format (RTF)** – This format displays colors, different fonts, emphasis, and formatting, such as bullets, text alignment, and linked objects.
- **HTML** – This format is sent as an HTML page, complete with tags to change the appearance of the text. The **recipient's (1)** e-mail client program then formats and displays the HTML.

1.4 Relationship to Protocols and Other Algorithms

This algorithm relies on [MS-OXCROPS], [MS-OXPROPS], and [MS-OXCDATA] for the specification of the **RopGetPropertiesSpecific ROP request** ([MS-OXCROPS] section 2.2.8.3), property values, and status codes.

1.5 Applicability Statement

The algorithm described in this document is used by a client to determine the format in which to retrieve a message from the server, when the client accepts multiple message body (2) formats.

The best body algorithm applies to **Message objects** of all types except when the following conditions are true:

- The value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) is exactly "IPM.Note.SMIME" (section 2.1.3.3). If the value of the **PidTagMessageClass** property is "IPM.Note.SMIME.MultipartSigned", then the algorithm described in section 2.1.3.1 is applicable.
- The value of the **PidTagMessageClass** property is "IPM.Note" and the value of the **PidNameContentClass** property ([MS-OXPROPS] section 2.435) is "rpmsg.Message", as described in section 2.1.3.4.

1.6 Standards Assignments

None.

2 Algorithm Details

2.1 Best Body Determination Algorithm Details

This section specifies the algorithm that determines the format of the message body (2) that is most like the original message, in order to maintain the richness of the text and formatting in the original message.

2.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this algorithm. The described organization is provided to facilitate the explanation of how the algorithm 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.

This algorithm includes the following ADM types, which are directly accessed from the Message and Attachment Object Protocol Specification as specified in [MS-OXCMSG] section 3.1.1:

Mailbox

Message Object

The following ADM elements are specific to this protocol:

MessageObject.BestBody: The original, primary, or best message body (2) format of the Message object.

2.1.2 Initialization

None.

2.1.3 Processing Rules

The processing rules for the best body algorithm as well as considerations for applying this algorithm to different message types are specified in sections 2.1.3.1 through 2.1.3.5.

2.1.3.1 Best Body Algorithm

The Best Body Algorithm specifies the algorithm that the client uses to determine the best body format of a message.

Step 1. Issue a **RopGetPropertiesSpecific** ROP request ([MS-OXCROPS] section 2.2.8.3) for five properties: **PidTagBody** ([MS-OXPROPS] section 2.696), **PidTagRtfCompressed** ([MS-OXPROPS] section 2.1012), **PidTagHtml** ([MS-OXPROPS] section 2.809), **PidTagRtfInSync** ([MS-OXPROPS] section 2.1013), and **PidTagNativeBody** ([MS-OXPROPS] section 2.881).

If the **RopGetPropertiesSpecific** ROP returns a status code that indicates a failure, then the body type is undefined and the algorithm exits.

If the client does not request all three of the body type properties (**PidTagBody**, **PidTagRtfCompressed**, and **PidTagHtml**), the server SHOULD<1> return the best value for the **PidTagNativeBody** property that fits one of the requested body types.

If the client retrieves all five property values and the value of the **PidTagNativeBody** property is as specified in the following table, then the server has already saved the best body format to use in the value of the **PidTagNativeBody** property. In this case, it is not necessary to perform the remainder of the algorithm specified in this section.

If the server returns a value for the **PidTagNativeBody** property, the following table identifies the best body format that corresponds to the value. If the **PidTagNativeBody** property is missing or the value is not provided in the following table, proceed to the remaining steps of the algorithm.

Value of the PidTagNativeBody property	Property Identifier	Body format
1	**PidTagBody**	Plain
3	**PidTagRtfCompressed**	RTF
3	**PidTagHtml**	HTML

If the server does not return the **PidTagNativeBody** property but does return the remaining four property values, then the **RopGetPropertiesSpecific** ROP returns a **StandardPropertyRow** structure ([MS-OXCDATA] section 2.8.1.1). If any of the four property values were not retrieved, then the **RopGetPropertiesSpecific** ROP returns a **FlaggedPropertyRow** structure ([MS-OXCDATA] section 2.8.1.2).

Step 2. Create four variables: *PlainStatus*, *RtfStatus*, *HtmlStatus*, and *RtfInSync*. Examine the returned property values and assign values to the corresponding variables as follows. In each case, if there is an error code, then the value for the variable is either NotFound or NotEnoughMemory.

- *PlainStatus* – If the **RopGetPropertiesSpecific** ROP returned a **StandardPropertyRow** structure, or the value of the **PidTagBody** property is a **PtypString** ([MS-OXCDATA] section 2.11.1), then assign the NoError error code to the *PlainStatus* variable; else copy the error code from the **FlaggedPropertyValue** structure to the *PlainStatus* variable.
- *RtfStatus* - If the **RopGetPropertiesSpecific** ROP returned a **StandardPropertyRow** structure, or the value of the **PidTagRtfCompressed** property is a **PtypBinary** ([MS-OXCDATA] section 2.11.1) value, then assign the NoError error code to the *RtfStatus* variable; else copy the error code from the **FlaggedPropertyValue** structure to the *RtfStatus* variable.
- *HtmlStatus* - If the **RopGetPropertiesSpecific** ROP returned a **StandardPropertyRow** structure, or the value of the **PidTagHtml** property is a **PtypBinary** value, then assign the NoError error code to the *HtmlStatus* variable; else copy the error code from the **FlaggedPropertyValue** structure to the *HtmlStatus* variable.
- *RtfInSync* - If the **RopGetPropertiesSpecific** ROP returned a **StandardPropertyRow** structure, or the value of the **PidTagRtfInSync** property is **PtypBoolean** ([MS-OXCDATA] section 2.11.1), then copy the **PtypBoolean** value to the *RtfInSync* variable; else assign FALSE to the *RtfInSync* variable.

Step 3. Determine the body format based on values of the four variables created in step 2. The following table can be implemented as an "if-then-else" chain, in exactly the order specified.

	PlainStatus	RtfStatus	HtmlStatus	RtfInSync	Body format
1	NotFound	NotFound	NotFound	Any	Undefined
2	NotEnoughMemory	NotFound	NotFound	Any	Plain text

	PlainStatus	RtfStatus	HtmlStatus	RtfInSync	Body format
3	NotEnoughMemory	NotEnoughMemory	NotFound	Any	RTF
4	NotEnoughMemory	NotEnoughMemory	NotEnoughMemory	True	RTF
5	NotEnoughMemory	NotEnoughMemory	NotEnoughMemory	False	HTML
6	Any	NoError or NotEnoughMemory	NoError or NotEnoughMemory	True	RTF
7	Any	NoError or NotEnoughMemory	NoError or NotEnoughMemory	False	HTML
8	NoError or NotEnoughMemory	NoError or NotEnoughMemory	Any	True	RTF
9.1	NoError or NotEnoughMemory	NoError or NotEnoughMemory	Any	False	Plain text
9.2	NotFound	NoError or NotEnoughMemory	NotFound	Any	RTF
9.3	NoError or NotEnoughMemory	NotFound	NotFound	Any	Plain text
9.4	NotFound	NotFound	NoError or NotEnoughMemory	Any	HTML
10	If no other case fits				Plain text

This table can be implemented by using the following pseudocode. Each row of the table is one clause of an "if-else-if" chain. Within a row, each column is ANDed together to form the condition of an "if" clause. If there is a case that is not defined, then the BodyFormat is plain text.

	Code to implement

	`If PidTagNativeBody <> NotFound Then`
	`  BodyFormat = PidTagNativeBody`
	`Else`

1	`If ((PlainStatus = NotFound) And`
	`  (RtfStatus = NotFound) And`
	`  (HtmlStatus = NotFound)) Then`
	`  BodyFormat = Undefined`

2	`ElseIf ((PlainStatus = NotEnoughMemory) And`
	`  (RtfStatus = NotFound) And`
	`  (HtmlStatus = NotFound)) Then`
	`  BodyFormat = Plain`

3	`ElseIf ((PlainStatus = NotEnoughMemory) And`
	`  (RtfStatus = NotEnoughMemory) And`
	`  (HtmlStatus = NotFound)) Then`

	Code to implement
	BodyFormat = Rtf
4	ElseIf ((PlainStatus = NotEnoughMemory) And (RtfStatus = NotEnoughMemory) And (HtmlStatus = NotEnoughMemory) And (RtfInSync = True)) Then BodyFormat = Rtf
5	ElseIf ((PlainStatus = NotEnoughMemory) And (RtfStatus = NotEnoughMemory) And (HtmlStatus = NotEnoughMemory) And (RtfInSync = False)) Then BodyFormat = Html
6	ElseIf ((RtfStatus = NoError or RtfStatus = NotEnoughMemory) And (HtmlStatus = NoError or HtmlStatus = NotEnoughMemory) And (RtfInSync = True)) Then BodyFormat = Rtf
7	ElseIf ((RtfStatus = NoError or RtfStatus = NotEnoughMemory) And (HtmlStatus = NoError or HtmlStatus = NotEnoughMemory) And (RtfInSync = False)) Then BodyFormat = Html
8	ElseIf ((PlainStatus = NoError or PlainStatus = NotEnoughMemory) And (RtfStatus = NoError or RtfStatus = NotEnoughMemory) And (RtfInSync = True)) Then BodyFormat = Rtf
9.1	ElseIf ((PlainStatus = NoError or PlainStatus = NotEnoughMemory) And (RtfStatus = NoError or RtfStatus = NotEnoughMemory) And (RtfInSync = False)) Then BodyFormat = Plain
9.2	ElseIf ((PlainStatus = NotFound) And (RtfStatus = NoError or RtfStatus = NotEnoughMemory) And (HtmlStatus = NotFound) Then BodyFormat = Rtf
9.3	ElseIf ((PlainStatus = NoError or PlainStatus = NotEnoughMemory) And (RtfStatus = NotFound) And (HtmlStatus = NotFound) Then BodyFormat = Plain
9.4	ElseIf ((PlainStatus = NotFound) And (RtfStatus = NotFound) And (HtmlStatus = NoError or HtmlStatus = NotEnoughMemory) Then BodyFormat = Html

	Code to implement
10	Else BodyFormat = Plain
	End If End If

2.1.3.2 Determining Whether Plain Text or HTML Was Converted to RTF

When the result of the best body algorithm, as specified in section 2.1.3.1, is RTF, it is possible to determine whether the RTF was generated from original plain text or HTML, as specified in [MS-OXRTFEX].

2.1.3.3 Special Considerations for S/MIME Secure Messages

The best body algorithm, as specified in section 2.1.3.1, yields an accurate result for a clear-signed **S/MIME (Secure/Multipurpose Internet Mail Extensions)** message, meaning the value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) is "IPM.Note.SMIME.MultipartSigned". However, the result of the best body algorithm is undefined for other types of S/MIME messages, for example, when the value of the **PidTagMessageClass** property is "IPM.Note.SMIME". For details about these message types, see [MS-OXOSMIME].

2.1.3.4 Special Considerations for Rights-Managed Secure Messages

For rights-managed secure messages, the message body (2) properties specified in this document do not contain the actual message body (2); instead, they contain boilerplate text intended for recipients (1) whose clients do not support rights-managed secure messages. The actual message body (2) resides in an attachment and is not accessible as a property of the Message object. To obtain the actual message body (2), a client MUST decrypt and parse the attachment, as specified in [MS-OXORMMS].

While the best body algorithm, as specified in section 2.1.3.1, yields a result for rights-managed secure messages, that result applies to the boilerplate text and not to the actual message body (2).

2.1.3.5 Obtaining the Best Body from the Server

If a message has already been saved to the server, then the client SHOULD download the five properties and check the **PidTagNativeBody** property<2> ([MS-OXPROPS] section 2.881) first. If the **PidTagNativeBody** property is set, then it is not necessary to perform the algorithm specified in section 2.1.3.1 to determine the best possible body.

3 Algorithm Examples

In the following example, a simple HTML message is sent to a server.

```
From: <user1@example.com>
To: <user2@example.com>
Subject: test HTML message
Date: Tue, 24 Jan 2006 01:58:57 -0800
MIME-Version: 1.0
Content-Type: text/html
Content-Transfer-Encoding: 7bit
Content-Class: urn:content-classes:message
Importance: normal

<HTML><BODY>Test message, <b>please</b> delete.</BODY></HTML>
```

The four property values of interest are returned from the server with the following values.

Property	Value
PidTagBody ([MS-OXPROPS] section 2.696)	error, NotEnoughMemory
PidTagHtml ([MS-OXPROPS] section 2.809)	<HTML><HEAD><meta HTTP-equiv="Content-Type" content="text/HTML; charset=iso-8859-1"></HEAD><BODY>Test message, please delete.</BODY></HTML>
PidTagRtfCompressed ([MS-OXPROPS] section 2.1012)	error, NotEnoughMemory
PidTagRtfInSync ([MS-OXPROPS] section 2.1013)	FALSE

The best body algorithm, as specified in section 2.1.3.1, creates the four variables shown in the following table.

Variable	Value
PlainStatus	NotEnoughMemory
RtfStatus	NotEnoughMemory
HtmlStatus	NoError
RtfInSync	FALSE

The best body algorithm uses the four newly created variables and matches clause 7, as specified in section 2.1.3.1.

	Code to implement
7	`ElseIf ((RtfStatus = NoError or RtfStatus = NotEnoughMemory) And (HtmlStatus = NoError or HtmlStatus = NotEnoughMemory) And (RtfInSync = False)) Then BodyFormat = Html`

And the result returned is HTML body format.

4 Security

4.1 Security Considerations for Implementers

None.

4.2 Index of Security Parameters

None.

5 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® Exchange Server 2003
- Microsoft® Exchange Server 2007
- Microsoft® Exchange Server 2010
- Microsoft® Exchange Server 15 Technical Preview
- Microsoft® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010
- Microsoft® Outlook® 15 Technical Preview

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. 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 that is 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 the product does not follow the prescription.

<1> Section 2.1.3.1: The **PidTagNativeBody** property ([MS-OXPROPS] section 2.881) is not supported by Exchange 2003 or Exchange 2007.

<2> Section 2.1.3.5: The **PidTagNativeBody** property ([MS-OXPROPS] section 2.881) is not supported by Exchange 2003 or Exchange 2007.

6 Change Tracking

This section identifies changes that were made to the [MS-OXBBODY] protocol document between the October 2011 and January 2012 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
5 Appendix A: Product Behavior	Added Exchange 15 Technical Preview and Outlook 15 Technical Preview to the list of applicable product versions.	Y	Content updated.

7 Index