

[MS-OXCFOLD]: Folder 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.mspix>). 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		Revised and edited technical content.
12/03/2008	1.03		Revised and edited technical content.
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated technical content and 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	5.0.1	Editorial	Revised and edited the technical content.
08/04/2010	6.0	Major	Significantly changed the technical content.
11/03/2010	7.0	Major	Significantly changed the technical content.
03/18/2011	7.0	No change	No changes to the meaning, language, and formatting of the technical content.
08/05/2011	8.0	Major	Significantly changed the technical content.
10/07/2011	9.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction	7
1.1 Glossary	7
1.2 References	8
1.2.1 Normative References	8
1.2.2 Informative References	9
1.3 Overview	9
1.4 Relationship to Other Protocols	9
1.5 Prerequisites/Preconditions	10
1.6 Applicability Statement	10
1.7 Versioning and Capability Negotiation	10
1.8 Vendor-Extensible Fields	10
1.9 Standards Assignments	10
2 Messages	11
2.1 Transport	11
2.2 Message Syntax	11
2.2.1 ROPs	11
2.2.1.1 RopOpenFolder ROP	11
2.2.1.1.1 RopOpenFolder ROP Request Buffer	11
2.2.1.1.2 RopOpenFolder ROP Response Buffer	12
2.2.1.2 RopCreateFolder ROP	12
2.2.1.2.1 RopCreateFolder ROP Request Buffer	12
2.2.1.2.2 RopCreateFolder ROP Response Buffer	13
2.2.1.3 RopDeleteFolder ROP	14
2.2.1.3.1 RopDeleteFolder ROP Request Buffer	14
2.2.1.3.2 RopDeleteFolder ROP Response Buffer	15
2.2.1.4 RopSetSearchCriteria ROP	15
2.2.1.4.1 RopSetSearchCriteria ROP Request Buffer	15
2.2.1.4.2 RopSetSearchCriteria ROP Response Buffer	17
2.2.1.5 RopGetSearchCriteria ROP	17
2.2.1.5.1 RopGetSearchCriteria ROP Request Buffer	17
2.2.1.5.2 RopGetSearchCriteria ROP Response Buffer	18
2.2.1.6 RopMoveCopyMessages ROP	19
2.2.1.6.1 RopMoveCopyMessages ROP Request Buffer	19
2.2.1.6.2 RopMoveCopyMessages ROP Response Buffer	19
2.2.1.7 RopMoveFolder ROP	20
2.2.1.7.1 RopMoveFolder ROP Request Buffer	20
2.2.1.7.2 RopMoveFolder ROP Response Buffer	20
2.2.1.8 RopCopyFolder ROP	21
2.2.1.8.1 RopCopyFolder ROP Request Buffer	21
2.2.1.8.2 RopCopyFolder ROP Response Buffer	21
2.2.1.9 RopEmptyFolder ROP	22
2.2.1.9.1 RopEmptyFolder ROP Request Buffer	22
2.2.1.9.2 RopEmptyFolder ROP Response Buffer	22
2.2.1.10 RopHardDeleteMessagesAndSubfolders ROP	22
2.2.1.10.1 RopHardDeleteMessagesAndSubfolders ROP Request Buffer	22
2.2.1.10.2 RopHardDeleteMessagesAndSubfolders ROP Response Buffer	23
2.2.1.11 RopDeleteMessages ROP	23
2.2.1.11.1 RopDeleteMessages ROP Request Buffer	23
2.2.1.11.2 RopDeleteMessages ROP Response Buffer	24

2.2.1.12	RopHardDeleteMessages ROP	24
2.2.1.12.1	RopHardDeleteMessages ROP Request Buffer	24
2.2.1.12.2	RopHardDeleteMessages ROP Response Buffer	24
2.2.1.13	RopGetHierarchyTable ROP	25
2.2.1.13.1	RopGetHierarchyTable ROP Request Buffer	25
2.2.1.13.2	RopGetHierarchyTable ROP Response Buffer	26
2.2.1.14	RopGetContentsTable ROP	26
2.2.1.14.1	RopGetContentsTable ROP Request Buffer	26
2.2.1.14.2	RopGetContentsTable ROP Response Buffer	27
2.2.2	Folder Object Properties	27
2.2.2.1	General Properties	28
2.2.2.2	Folder Object Specific Properties.....	28
2.2.2.2.1	Read-Only Properties.....	28
2.2.2.2.1.1	PidTagContentCount Property.....	28
2.2.2.2.1.2	PidTagContentUnreadCount Property	28
2.2.2.2.1.3	PidTagDeletedOn Property	28
2.2.2.2.1.4	PidTagAddressBookEntryId Property	28
2.2.2.2.1.5	PidTagFolderId Property	28
2.2.2.2.1.6	PidTagHierarchyChangeNumber Property.....	29
2.2.2.2.1.7	PidTagMessageSize Property	29
2.2.2.2.1.8	PidTagMessageSizeExtended Property	29
2.2.2.2.1.9	PidTagSubfolders Property	29
2.2.2.2.2	Read/Write Properties.....	29
2.2.2.2.2.1	PidTagAttributeHidden Property.....	29
2.2.2.2.2.2	PidTagComment Property	29
2.2.2.2.2.3	PidTagDisplayName Property	29
2.2.2.2.2.4	PidTagFolderType Property	30
2.2.2.2.2.5	PidTagRights Property	30
2.2.2.2.2.6	PidTagAccessControllistData Property.....	30
3	Protocol Details.....	31
3.1	Client Details.....	31
3.1.1	Abstract Data Model	31
3.1.1.1	Hierarchy Table	31
3.1.1.2	Contents Table	31
3.1.2	Timers	32
3.1.3	Initialization	32
3.1.4	Higher-Layer Triggered Events.....	32
3.1.4.1	Opening a Folder	32
3.1.4.2	Creating a Folder	32
3.1.4.3	Deleting a Folder	32
3.1.4.4	Setting Up a Search Folder	32
3.1.4.5	Getting Details About a Search Folder	33
3.1.4.6	Moving a Folder and Its Contents.....	33
3.1.4.7	Copying a Folder and Its Contents	33
3.1.4.8	Deleting the Contents of a Folder.....	34
3.1.4.9	Getting a Hierarchy Table	34
3.1.4.10	Getting a Contents Table	34
3.1.5	Message Processing Events and Sequencing Rules.....	35
3.1.5.1	Releasing a Server Object.....	35
3.1.5.2	Processing ROPs Asynchronously	35
3.1.6	Timer Events	35
3.1.7	Other Local Events	35

3.2	Server Details	35
3.2.1	Abstract Data Model	35
3.2.2	Timers	36
3.2.3	Initialization	36
3.2.4	Higher-Layer Triggered Events.....	36
3.2.5	Message Processing Events and Sequencing Rules.....	36
3.2.5.1	Processing a RopOpenFolder ROP Request	36
3.2.5.2	Processing a RopCreateFolder ROP Request	37
3.2.5.3	Processing a RopDeleteFolder ROP Request.....	37
3.2.5.4	Processing a RopSetSearchCriteria ROP Request.....	38
3.2.5.5	Processing a RopGetSearchCriteria ROP Request	40
3.2.5.6	Processing a RopMoveCopyMessages ROP Request	40
3.2.5.7	Processing a RopMoveFolder ROP Request	41
3.2.5.8	Processing a RopCopyFolder ROP Request.....	41
3.2.5.9	Processing a RopEmptyFolder ROP Request.....	42
3.2.5.10	Processing a RopHardDeleteMessagesAndSubfolders ROP Request	43
3.2.5.11	Processing a RopDeleteMessages ROP Request	43
3.2.5.12	Processing a RopHardDeleteMessages ROP Request	44
3.2.5.13	Processing a RopGetHierarchyTable ROP Request.....	44
3.2.5.14	Processing a RopGetContentsTable ROP Request.....	45
3.2.6	Timer Events	45
3.2.7	Other Local Events	45
4	Protocol Examples.....	46
4.1	Creating a New Folder	46
4.1.1	Client Request Buffer	46
4.1.2	Server Responds to Client Request	47
4.2	Deleting an Existing Folder.....	48
4.2.1	Client Request Buffer	48
4.2.2	Server Responds to Client Request	49
4.3	Deleting Messages Within a Folder	49
4.3.1	Client Request Buffer	49
4.3.2	Server Responds to Client Request	50
4.4	Moving Messages from One Folder to Another	50
4.4.1	Client Request Buffer	50
4.4.2	Server Responds to Client Request	51
4.5	Moving a Folder	51
4.5.1	Client Request Buffer	52
4.5.2	Server Responds to Client Request	52
4.6	Copying a Folder.....	53
4.6.1	Client Request Buffer	53
4.6.2	Server Responds to Client Request	54
4.7	Getting the List of Subfolders Within a Message Folder	54
4.7.1	Client Request Buffer	54
4.7.2	Server Responds to Client Request	54
4.8	Setting the Search Criteria for a Search Folder	55
4.8.1	Client Request Buffer	55
4.8.2	Server Responds to Client Request	58
4.9	Getting the Search Criteria for a Search Folder	59
4.9.1	Client Request Buffer	59
4.9.2	Server Responds to Client Request	59
5	Security.....	64

5.1 Security Considerations for Implementers	64
5.2 Index of Security Parameters	64
6 Appendix A: Product Behavior	65
7 Change Tracking.....	67
8 Index	69

1 Introduction

The Folder Object Protocol enables a client to create a folder and to manipulate an existing folder and its contents, which can include messages and subfolders.

Sections 1.8, 2, and 3 of this specification are normative and contain RFC 2119 language. Sections 1.5 and 1.9 are 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 [\[MS-GLOS\]](#):

- access control list (ACL)**
- ASCII**
- code page**
- handle**
- little-endian**
- remote procedure call (RPC)**
- Unicode**

The following terms are defined in [\[MS-OXGLOS\]](#):

- active replica**
- contents table**
- Drafts folder**
- entry ID**
- EntryID**
- folder associated information (FAI)**
- Folder object**
- ghosted folder**
- hard delete**
- hierarchy table**
- Inbox folder**
- Logon object**
- mailbox**
- Message object**
- messaging object**
- non-read receipt**
- normal message**
- permission**
- property type**
- public folder**
- read receipt**
- remote operation (ROP)**
- replica**
- restriction**
- Root folder**
- ROP buffer**
- ROP request**
- ROP request buffer**
- ROP response**
- ROP response buffer**
- rule**
- search criteria**

search folder
search folder container
Server object
Server object handle
Server object handle table
soft delete
special folder
store
stream
Table object

The following terms are specific to this document:

full-text index: A digitally stored list of search terms that is culled by examining all the content in the bodies of documents, messages, or other text objects, in order to increase the speed of search results.

full-text search: In text retrieval, a technique for searching a computer-stored document or database by examining all the words in every stored document, and attempting to match the search words supplied by the client.

sibling folder: A name that is given to two or more generic folders that have the same parent folder.

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-OXCDATA] Microsoft Corporation, "[Data Structures](#)".

[MS-OXCFXICS] Microsoft Corporation, "[Bulk Data Transfer Protocol Specification](#)".

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

[MS-OXCNOTIF] Microsoft Corporation, "[Core Notifications Protocol Specification](#)".

[MS-OXCPerm] Microsoft Corporation, "[Exchange Access and Operation Permissions Protocol Specification](#)".

[MS-OXCPRPT] Microsoft Corporation, "[Property and Stream Object Protocol Specification](#)".

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

[MS-OXCRPC] Microsoft Corporation, "[Wire Format Protocol Specification](#)".

[MS-OXCSTOR] Microsoft Corporation, "[Store Object Protocol Specification](#)".

[MS-OXCTABL] Microsoft Corporation, "[Table Object Protocol Specification](#)".

[MS-OXORULE] Microsoft Corporation, "[E-Mail Rules Protocol Specification](#)".

[MS-OXOSFLD] Microsoft Corporation, "[Special Folders Protocol Specification](#)".

[MS-OXOSRCH] Microsoft Corporation, "[Search Folder List Configuration Protocol Specification](#)".

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

[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-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

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

[MS-OXOMSG] Microsoft Corporation, "[E-Mail Object Protocol Specification](#)".

1.3 Overview

A folder is the basic unit of organization for **messaging objects** in a **store**. A folder is represented in the store by a **Folder object**. This protocol enables a client to create folders and to manipulate existing folders and their contents by using **remote operations (ROPs)**. A client can also modify the **permissions** on a folder. For information about folder permissions, see [\[MS-OXCPERM\]](#).

Folders are arranged hierarchically. Each folder has properties associated with it. When a folder is opened, the Folder object that is returned by a ROP can then be used in subsequent **ROP requests**. The ROPs for a Folder object are described in section [2.2.1](#). The properties of a Folder object are described in section [2.2.2](#).

The following are the three types of folders:

- **Root folder**. Every store has a Root folder. The Root folder appears at the top of the folder hierarchy and can contain messages and other folders. The Root folder cannot be moved, copied, renamed, or deleted. There is only one Root folder for each store.
- **Generic folder**. Like a Root folder, a generic folder can contain messages and other folders. Unlike a Root folder, a generic folder can be moved, copied, renamed, and deleted. A generic folder can be created within either the Root folder or another generic folder. The folder in which a folder is created is referred to as the parent folder of the new folder. Generic folders that have the same parent are called **sibling folders**.
- **Search folder (2)**. A search folder (2) contains a list of references to messages. The list is compiled by the server according to a set of criteria. Therefore, a search folder (2) cannot contain any real objects. Any operation on a message that is referenced in a search folder (2) is performed on the message in the folder that actually contains the message. For more information about search folders (2), see [\[MS-OXOSRCH\]](#).

1.4 Relationship to Other Protocols

The Folder Object Protocol uses other protocols as follows:

- The Remote Operations (ROP) List and Encoding Protocol, described in [\[MS-OXCROPS\]](#), to format the **ROP buffers** for transmission between client and server.

- The Wire Format Protocol, described in [\[MS-OXCRPC\]](#), to transmit the ROP buffers between client and server.
- The the Store Object Protocol, described in [\[MS-OXCSTOR\]](#), to log on to the store.
- The Property and Stream Object Protocol, described in [\[MS-OXCPRPT\]](#), to set properties on a Folder object.
- The Table Object Protocol, described in [\[MS-OXCTABL\]](#), to manipulate the **Table objects** that are retrieved by the **RopGetHierarchyTable** (section [2.2.1.13](#)) and **RopGetContentsTable** (section [2.2.1.14](#)) ROPs.
- The Exchange Access and Operation Permissions Protocol, described in [\[MS-OXCPERM\]](#), to retrieve and set permissions on a Folder object.

The following protocols extend the Folder Object Protocol:

- Search Folder List Configuration Protocol, described in [\[MS-OXOSRCH\]](#).
- Special Folders Protocol, described in [\[MS-OXOSFLD\]](#).

1.5 Prerequisites/Preconditions

This protocol assumes that the client has previously logged on to the store, as specified in [\[MS-OXCSTOR\]](#) section 3.1.4.1, and has acquired a **handle** to the **Server object** on which it is going to operate.

1.6 Applicability Statement

This protocol is applicable when the client needs to create a folder in a store or to operate on an existing folder in a store. This protocol also applies when the client needs to manipulate a folder's contents, which can include messages and subfolders.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol uses the ROP List and Encoding Protocol, specified in [\[MS-OXCROPS\]](#), to format the **ROP request buffers** and **ROP response buffers**. These buffers are then transported between the client and the server by using the Wire Format Protocol, as specified in [\[MS-OXCRPC\]](#).

2.2 Message Syntax

2.2.1 ROPs

The format of ROP request buffers and ROP response buffers that are specific to folder operations is specified in sections [2.2.1.1](#) through [2.2.1.14](#). For ROPs that require a folder identifier or a message identifier, the client needs to acquire those identifiers for the objects to be used in the ROP requests. For more details about acquiring message identifiers, including usages, **restrictions (2)**, and notes, see [\[MS-OXCMSG\]](#).

Some fields are not included in the ROP specifications. The descriptions of ROP request buffers do not include the **RopId** field and the **LogonId** field. The descriptions of the ROP response buffers do not include the **RopId** field and the fields that specify handle indexes. For details about these fields, see [\[MS-OXCROPS\]](#) section 2.2.4.

2.2.1.1 RopOpenFolder ROP

The **RopOpenFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.1) opens an existing folder.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.1.1 RopOpenFolder ROP Request Buffer

The following descriptions define valid fields for the **RopOpenFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.1.1).

InputHandleIndex (1 byte): An integer that specifies the location in the **Server object handle table** where the handle for the input Server object is stored. The input Server object for this operation is a **Logon object** or a Folder object that represents the object to be opened. For details about Logon objects, see [\[MS-OXCSTOR\]](#).

OutputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the output Server object is stored. The output Server object for this operation is a Folder object that represents the folder that was opened.

FolderId (8 bytes): A **Folder ID (FID)** structure ([\[MS-OXCDATA\]](#) section 2.2.1.1) that specifies the folder to be opened.

OpenModeFlags (1 byte): A set of bits that indicate the mode for opening the folder.

The valid bits for this field are specified in the following table. All other bits MUST NOT be set by the client and MUST be ignored by the server.

Bit name	Value	Meaning
OpenSoftDeleted	0x04	If this bit is set, the operation opens either an existing folder or a soft-deleted folder. If this bit is not set, the operation opens an existing folder.

2.2.1.1.2 RopOpenFolder ROP Response Buffer

The following descriptions define valid fields for the **RopOpenFolder** ROP response buffer ([MS-OXCROPS] section 2.2.4.1.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [MS-OXCADATA] section 2.4.

HasRules (1 byte): A Boolean value that indicates whether **rules (4)** are associated with the folder. If rules (4) are associated with the folder, this field is set to a nonzero (TRUE) value; otherwise, this field is set to zero (FALSE). <1> For details about rules (4), see [MS-OXORULE].

IsGhosed (1 byte): A Boolean value that indicates whether the server hosts an **active replica** of the folder. If the server does not host an active replica of the folder, this field is set to a nonzero (TRUE) value; otherwise, this field is set to zero (FALSE). This field is present only for folders that are in a public store.

ServerCount (2 bytes): An integer that specifies the number of servers that have a **replica** of the folder. This field is present only if the **IsGhosed** field is set to a nonzero (TRUE) value.

CheapServerCount (2 bytes): An integer that specifies the number of the cheapest, same-cost servers at the front of the server list. The value of this field MUST be less than or equal to the value of the **ServerCount** field. This field is present only if the **IsGhosed** field is set to a nonzero (TRUE) value.

Servers (variable): An array of null-terminated **ASCII** strings, each of which specifies a server that has a replica of the folder. The number of strings contained in this field is specified by the **ServerCount** field. This field is present only if the **IsGhosed** field is set to a nonzero (TRUE) value.

2.2.1.2 RopCreateFolder ROP

The **RopCreateFolder** ROP ([MS-OXCROPS] section 2.2.4.2) creates a new folder. The folder can be either a **public folder** or a private **mailbox** folder.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [MS-OXCROPS]. This section specifies the syntax and semantics of various fields that are not fully specified in [MS-OXCROPS].

2.2.1.2.1 RopCreateFolder ROP Request Buffer

The following descriptions define valid fields for the **RopCreateFolder** ROP request buffer ([MS-OXCROPS] section 2.2.4.2.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the parent folder of the folder to be created.

OutputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the output Server object is stored. The output Server object for this operation is a Folder object that represents the folder that was created.

FolderType (1 byte): An integer that specifies the type of folder to be created. One of the values specified in the following table MUST be used.

Value	Folder type
1	Generic folder
2	Search folder

UseUnicodeStrings (1 byte): A Boolean value that is nonzero (TRUE) if the values of the the **DisplayName** and **Comment** fields are formatted in **Unicode** and zero (FALSE) otherwise.

OpenExisting (1 byte): A Boolean value that is nonzero (TRUE) if a pre-existing folder, whose name is identical to the name specified in the **DisplayName** field, is to be opened and zero (FALSE) otherwise.

Reserved (1 byte): This field is reserved. The client MUST set this field to zero (FALSE).

DisplayName (variable): A null-terminated string that specifies the display name of the folder. This name becomes the value of the new folder's **PidTagDisplayName** property (section [2.2.2.2.2.3](#)).

Comment (variable): A null-terminated folder string that specifies a comment associated with the new folder. The comment can be used to describe the folder. This string becomes the value of the new folder's **PidTagComment** property (section [2.2.2.2.2.2](#)).

2.2.1.2.2 RopCreateFolder ROP Response Buffer

The following descriptions define valid fields for the **RopCreateFolder** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.2.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCADATA\]](#) section 2.4.

FolderId (8 bytes): A **Folder ID (FID)** structure ([\[MS-OXCADATA\]](#) section 2.2.1.1) that specifies the folder that was created or opened.

IsExistingFolder (1 byte): A Boolean value that is nonzero (TRUE) if a folder with the name given by the **DisplayName** field of the request buffer (section [2.2.1.2.1](#)) already exists. [<2>](#) The value is zero (FALSE) if the folder with that name does not exist.

HasRules (1 byte): A Boolean value that indicates whether rules (4) are associated with the folder. If rules (4) are associated with the folder, this field is set to a nonzero (TRUE) value; otherwise, this field is set to zero (FALSE). [<3>](#) For details about rules (4), see [\[MS-OXORULE\]](#). This field is present only if the **IsExistingFolder** field is set to a nonzero (TRUE) value.

IsGhosted (1 byte): A Boolean value that indicates whether the server hosts an active replica of the folder. (If the folder is not an active replica, it is a **ghosted folder**.) If the server does not host an active replica of the folder, this field is set to a nonzero (TRUE) value; otherwise,

this field is set to zero (FALSE). This field is present only if the **IsExistingFolder** field is set to a nonzero (TRUE) value and only for folders that are in a public store.

ServerCount (2 bytes): An integer that specifies the number of servers that have a replica of the folder. This field is present only if the **IsGhosted** field is set to a nonzero (TRUE) value.

CheapServerCount (2 bytes): An integer that specifies the number of the cheapest, same-cost servers. These servers are listed at the beginning of the array contained in the **Servers** field. The value of this field MUST be less than or equal to the value of the **ServerCount** field and MUST be greater than zero when the value of the **ServerCount** field is greater than zero. This field is present only if the **IsGhosted** field is set to a nonzero (TRUE) value.

Servers (variable): An array of null-terminated ASCII strings, each of which specifies a server that has a replica of the folder. The number of strings contained in this field is specified by the **ServerCount** field. This field is present only if the **IsGhosted** field is set to a nonzero (TRUE) value.

2.2.1.3 RopDeleteFolder ROP

The **RopDeleteFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.3) removes a folder. By default, the **RopDeleteFolder** ROP operates only on empty folders.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.3.1 RopDeleteFolder ROP Request Buffer

The following descriptions define valid fields for the **RopDeleteFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.3.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the parent folder of the folder to be deleted.

DeleteFolderFlags (1 byte): A set of bits that control the deletion of a folder. By default, the **RopDeleteFolder** ROP operates only on empty folders, but it can be used successfully on non-empty folders by setting the DEL_FOLDERS bit and the DEL_MESSAGES bit.

The valid bits for this field are listed in the following table. The client MUST NOT set any other bits.

Bit name	Value	Meaning
DEL_MESSAGES	0x01	The folder and all of the Message objects in the folder are deleted.
DEL_FOLDERS	0x04	The folder and all of its subfolders are deleted.
DELETE_HARD_DELETE	0x10	If this bit is set, the folder is hard deleted . If it is not set, the folder is soft deleted .

FolderId (8 bytes): A **FID** structure ([\[MS-OXCDATA\]](#) section 2.2.1.1) that specifies the folder to be deleted.

2.2.1.3.2 RopDeleteFolder ROP Response Buffer

The following descriptions define valid fields for the **RopDeleteFolder** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.3.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCDATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE); otherwise, the value is zero (FALSE).

2.2.1.4 RopSetSearchCriteria ROP

The **RopSetSearchCriteria** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.4) establishes **search criteria** for a search folder (2).

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.4.1 RopSetSearchCriteria ROP Request Buffer

The following descriptions define valid fields for the **RopSetSearchCriteria** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.4.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the search folder (2).

RestrictionDataSize (2 bytes): An integer that specifies the size of the **RestrictionData** field. If the value of the **RestrictionDataSize** field is zero, the search criteria that were used most recently for the **search folder container** are used again. The **RestrictionDataSize** field MUST NOT be set to zero for the first search.

RestrictionData (variable): A packet of structures that specify restrictions (2) for the search folder (2). The size of this field is specified by the value of the **RestrictionDataSize** field. For details about the structures that are used to specify restrictions (2), see [\[MS-OXCDATA\]](#) section 2.12.

FolderIdCount (2 bytes): An integer that specifies the number of structures contained in the **FolderIds** field. If the **FolderIdCount** field is set to zero, the folders that were used in the most recent search are used again. The **FolderIdCount** field MUST NOT be set to zero for the first search within a search folder container.

FolderIds (variable): An array of **FID** structures ([\[MS-OXCDATA\]](#) section 2.2.1.1), each of which specifies a folder that will be searched. The number of structures contained in the array is specified by the value of the **FolderIdCount** field.

SearchFlags (4 bytes): A set of bits that control the search for a search folder (2). For more details about how these bits affect the search, see section [3.2.5.4](#).

The valid bits for this field are listed in the following table. The client MUST NOT set any other bits.

Bit name	Value	Meaning
STOP_SEARCH	0x00000001	The search is aborted. This bit MUST NOT be set at the same time as the RESTART_SEARCH bit.
RESTART_SEARCH	0x00000002	The search is initiated if this is the first RopSetSearchCriteria ROP request, or if the search is restarted, or if the search is inactive. This bit MUST NOT be set at the same time as the STOP_SEARCH bit.
RECURSIVE_SEARCH	0x00000004	The search includes the search folder containers and all of their child folders. This bit MUST NOT be set at the same time as the SHALLOW_SEARCH bit. If neither the RECURSIVE_SEARCH bit nor the SHALLOW_SEARCH bit is set, the RECURSIVE_SEARCH and SHALLOW_SEARCH settings from the previous RopSetSearchCriteria ROP request will be used. If neither the RECURSIVE_SEARCH bit nor the SHALLOW_SEARCH bit is set in the first RopSetSearchCriteria ROP request, the settings default to SHALLOW_SEARCH.
SHALLOW_SEARCH	0x00000008	The search includes only the search folder containers that are specified in the FolderIds field. This bit MUST NOT be set at the same time as the RECURSIVE_SEARCH bit. If neither the RECURSIVE_SEARCH bit nor the SHALLOW_SEARCH bit is set, the RECURSIVE_SEARCH and SHALLOW_SEARCH settings from the previous RopSetSearchCriteria ROP request will be used. If neither the RECURSIVE_SEARCH bit nor the SHALLOW_SEARCH bit is set in the first RopSetSearchCriteria ROP request, the settings default to SHALLOW_SEARCH.
FOREGROUND_SEARCH	0x00000010	The search is executed at a high priority relative to other searches. This bit MUST NOT be set at the same time as the BACKGROUND_SEARCH bit. If neither the FOREGROUND_SEARCH bit nor the BACKGROUND_SEARCH bit is set, the FOREGROUND_SEARCH and BACKGROUND_SEARCH settings from the previous RopSetSearchCriteria ROP request will be used. If neither the FOREGROUND_SEARCH bit nor the BACKGROUND_SEARCH bit is set in the first RopSetSearchCriteria ROP request, the settings default to BACKGROUND_SEARCH.
BACKGROUND_SEARCH	0x00000020	The search is executed at normal priority relative to other searches. This bit MUST NOT be set at the same time as the FOREGROUND_SEARCH bit. If neither the FOREGROUND_SEARCH bit nor the BACKGROUND_SEARCH bit is set, the

Bit name	Value	Meaning
		BACKGROUND_SEARCH and FOREGROUND_SEARCH settings from the previous RopSetSearchCriteria ROP request will be used. If neither the FOREGROUND_SEARCH bit nor the BACKGROUND_SEARCH bit is set in the first RopSetSearchCriteria ROP request, the settings default to BACKGROUND_SEARCH.
CONTENT_INDEXED_SEARCH	0x00010000	The search uses a content-indexed search.
NON_CONTENT_INDEXED_SEARCH	0x00020000	The search does not use a content-indexed search.
STATIC_SEARCH	0x00040000	The search is static.

2.2.1.4.2 RopSetSearchCriteria ROP Response Buffer

The following descriptions define valid fields for the **RopSetSearchCriteria** ROP response buffer ([MS-OXCROPS] section 2.2.4.4.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [MS-OXCDATA] section 2.4.

2.2.1.5 RopGetSearchCriteria ROP

The **RopGetSearchCriteria** ROP ([MS-OXCROPS] section 2.2.4.5) obtains the search criteria and the status of a search for a search folder (2). The search criteria are created by using **RopSetSearchCriteria** (section 2.2.1.4).

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [MS-OXCROPS]. This section specifies the syntax and semantics of various fields that are not fully specified in [MS-OXCROPS].

2.2.1.5.1 RopGetSearchCriteria ROP Request Buffer

The following descriptions define valid fields for the **RopGetSearchCriteria** ROP request buffer ([MS-OXCROPS] section 2.2.4.5.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the search folder (2).

UseUnicode (1 byte): A Boolean value that is nonzero (TRUE) if the value of the **RestrictionData** field of the **ROP response** is to be in Unicode format or zero (FALSE) otherwise.

IncludeRestriction (1 byte): A Boolean value that is nonzero (TRUE) if the restriction (2) data is required in the response or zero (FALSE) otherwise.

IncludeFolders (1 byte): A Boolean value that is nonzero (TRUE) if the list of folders being searched is required in the response or zero (FALSE) otherwise.

2.2.1.5.2 RopGetSearchCriteria ROP Response Buffer

The following descriptions define valid fields for the **RopGetSearchCriteria** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.5.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCDATA\]](#) section 2.4.

RestrictionDataSize (2 bytes): An integer that specifies the size, in bytes, of the **RestrictionData** field. If the **IncludeRestriction** field of the request buffer was set to zero (FALSE), the value of **RestrictionDataSize** will be 0.

RestrictionData (variable): A packet of structures that specify restrictions (2) for the search folder (2). For details about the structures that are used to specify restrictions (2), see [\[MS-OXCDATA\]](#) section 2.12. The size of this field is specified by the **RestrictionDataSize** field. This field is present only if the value of the **RestrictionDataSize** field is nonzero (TRUE).

FolderIdCount (2 bytes): An integer that specifies the number of structures contained in the **FolderIds** field. If the **IncludeFolders** field of the request buffer was set to zero (FALSE), the **FolderIdCount** field will be set to 0.

FolderIds (variable): An array of **FID** structures ([\[MS-OXCDATA\]](#) section 2.2.1.1), each of which specifies a folder that is being searched. The number of structures contained in the array is specified by the value of the **FolderIdCount** field. This field is present only if the value of the **FolderIdCount** field is nonzero (TRUE).

SearchFlags (4 bytes): A set of bits that indicate the state of the current search.

The valid bits for this field are listed in the following table. The client MUST ignore any other bits.

Bit name	Value	Meaning
SEARCH_RUNNING	0x00000001	The search is running, which means that the initial population of the search folder (2) still being compiled.
SEARCH_REBUILD	0x00000002	The search is in the CPU-intensive part of the search. This bit is set only if the SEARCH_RUNNING bit is also set.
SEARCH_RECURSIVE	0x00000004	If this bit is set, the specified search folder containers and all their child search folder containers are searched for matching entries. If this bit is not set, only the search folder containers that are specified in the last RopSetSearchCriteria ROP request (section 2.2.1.4.1) are being searched.
SEARCH_FOREGROUND	0x00000008	If this bit is set, the search is running at a high priority relative to other searches. If this bit is not set, the search is running at a normal priority relative to other searches.
SEARCH_COMPLETE	0x00001000	The search results are complete.
SEARCH_PARTIAL	0x00002000	Only some parts of messages were included.
SEARCH_STATIC	0x00010000	The search is static.

Bit name	Value	Meaning
SEARCH_MAYBE_STATIC	0x00020000	The search is still being evaluated.
CI_TOTALLY	0x01000000	The search is done using content indexing.
TWIR_TOTALLY	0x08000000	The search is done using store-only search.

2.2.1.6 RopMoveCopyMessages ROP

The **RopMoveCopyMessages** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.6) moves or copies messages from a source folder to a destination folder. This ROP applies to both public folders and private mailboxes.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.6.1 RopMoveCopyMessages ROP Request Buffer

The following descriptions define valid fields for the **RopMoveCopyMessages** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.6.1).

SourceHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the source Server object is stored. The source Server object for this operation is a Folder object that represents the folder from which the messages will be moved or copied.

DestHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the destination Server object is stored. The destination Server object for this operation is a Folder object that represents the folder to which the messages will be moved or copied.

MessageIdCount (2 bytes): An integer that specifies the number of structures contained in the **MessageIds** field.

MessageIds (variable): An array of **Message ID (MID)** structures ([\[MS-OXCADATA\]](#) section 2.2.1.2), each of which specifies a message to be moved or copied. The number of structures contained in the array is specified by the value of the **MessageIdCount** field.

WantAsynchronous (1 byte): A Boolean value that is nonzero (TRUE) if the ROP is to be processed asynchronously or zero (FALSE) if the ROP is to be processed synchronously. For details about asynchronous processing, see section [3.1.5.2](#).

WantCopy (1 byte): A Boolean value that is nonzero (TRUE) if this is a copy operation or zero (FALSE) if this is a move operation.

2.2.1.6.2 RopMoveCopyMessages ROP Response Buffer

The following descriptions define valid fields for the **RopMoveCopyMessages** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.6.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCADATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE). Otherwise, the value is zero (FALSE).

2.2.1.7 RopMoveFolder ROP

The **RopMoveFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.7) moves a folder from one parent folder to another parent folder. All contents of the folder are moved with it. The move can be within a private mailbox or a public folder, or between a private mailbox and a public folder.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.7.1 RopMoveFolder ROP Request Buffer

The following descriptions define valid fields for the **RopMoveFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.7.1).

SourceHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the source Server object is stored. The source Server object for this operation is a Folder object that represents the parent folder from which the folder will be moved.

DestHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the destination Server object is stored. The destination Server object for this operation is a Folder object that represents the parent folder to which the folder will be moved.

WantAsynchronous (1 byte): A Boolean value that is nonzero (TRUE) if the ROP is to be processed asynchronously or zero (FALSE) if the ROP is to be processed synchronously. For details about asynchronous processing, see section [3.1.5.2](#).

UseUnicode (1 byte): A Boolean value that is nonzero (TRUE) if the value of the **NewFolderName** field is formatted in Unicode; it is zero (FALSE) otherwise.

FolderId (8 bytes): A **FID** structure ([\[MS-OXCADATA\]](#) section 2.2.1.1) that specifies the folder to be moved.

NewFolderName (variable): A null-terminated string that specifies the new name for the moved folder.

2.2.1.7.2 RopMoveFolder ROP Response Buffer

The following descriptions define valid fields for the **RopMoveFolder** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.7.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCADATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE). Otherwise, the value is zero (FALSE).

2.2.1.8 RopCopyFolder ROP

The **RopCopyFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.8) copies a folder from one parent folder to another parent folder. All contents of the folder are copied with it. The operation can be performed within a private mailbox or a public folder, or between a private mailbox and a public folder.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.8.1 RopCopyFolder ROP Request Buffer

The following descriptions define valid fields for the **RopCopyFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.8.1).

SourceHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the source Server object is stored. The source Server object for this operation is a Folder object that represents the parent folder from which the folder will be copied.

DestHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the destination Server object is stored. The destination Server object for this operation is a Folder object that represents the parent folder to which the folder will be copied.

WantAsynchronous (1 byte): A Boolean value that is nonzero (TRUE) if the ROP is to be processed asynchronously or zero (FALSE) if the ROP is to be processed synchronously. For details about asynchronous processing, see section [3.1.5.2](#).

WantRecursive (1 byte): A Boolean value that is nonzero (TRUE) if the folder is to be copied recursively—that is, all of the folder's subfolders are copied to the new folder and the subfolders' subfolders are copied to the new folder and so on. The value is zero (FALSE) otherwise.

UseUnicode (1 byte): A Boolean value that is nonzero (TRUE) if the value of the **NewFolderName** field is formatted in Unicode; it is zero (FALSE) otherwise.

FolderId (8 bytes): A **FID** structure ([\[MS-OXCADATA\]](#) section 2.2.1.1) that specifies the folder to be copied.

NewFolderName (variable): A null-terminated string that specifies the new name for the copied folder.

2.2.1.8.2 RopCopyFolder ROP Response Buffer

The following descriptions define valid fields for the **RopCopyFolder** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.8.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCADATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE). Otherwise, the value is zero (FALSE).

2.2.1.9 RopEmptyFolder ROP

The **RopEmptyFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.9) is used to soft delete messages and subfolders from a folder without deleting the folder itself. To hard delete all messages and subfolders from a folder, use the **RopHardDeleteMessagesAndSubfolders** ROP (section [2.2.1.10](#)).

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.9.1 RopEmptyFolder ROP Request Buffer

The following descriptions define valid fields for the **RopEmptyFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.9.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the folder whose messages and subfolders will be soft deleted.

WantAsynchronous (1 byte): A Boolean value that is nonzero (TRUE) if the ROP is to be processed asynchronously or zero (FALSE) if the ROP is to be processed synchronously. For details about asynchronous processing, see section [3.1.5.2](#).

WantDeleteAssociated (1 byte): A Boolean value that is nonzero (TRUE) if the **folder associated information (FAI)** messages are to be included in the deletion. The value is zero (FALSE) otherwise.

2.2.1.9.2 RopEmptyFolder ROP Response Buffer

The following descriptions define valid fields for the **RopEmptyFolder** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.9.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCADATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE). Otherwise, the value is zero (FALSE).

2.2.1.10 RopHardDeleteMessagesAndSubfolders ROP

The **RopHardDeleteMessagesAndSubfolders** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.10) is used to hard delete all messages and subfolders from a folder without deleting the folder itself.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.10.1 RopHardDeleteMessagesAndSubfolders ROP Request Buffer

The following descriptions define valid fields for the **RopHardDeleteMessagesAndSubfolders** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.10.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the folder whose messages and subfolders will be hard deleted.

WantAsynchronous (1 byte): A Boolean value that is nonzero (TRUE) if the ROP is to be processed asynchronously or zero (FALSE) if the ROP is to be processed synchronously. For details about asynchronous processing, see section [3.1.5.2](#).

WantDeleteAssociated (1 byte): A Boolean value that is nonzero (TRUE) if the FAI messages are to be included in the deletion. The value is zero (FALSE) otherwise.

2.2.1.10.2 RopHardDeleteMessagesAndSubfolders ROP Response Buffer

The following descriptions define valid fields for the **RopHardDeleteMessagesAndSubfolders** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.10.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCADATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE). Otherwise, the value is zero (FALSE).

2.2.1.11 RopDeleteMessages ROP

The **RopDeleteMessages** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.11) is used to soft delete one or more messages from a folder. A message that has been soft deleted can be recovered only by opening it, copying it to a new message, and then saving the new message. For details about opening, copying, and saving a Message object, see [\[MS-OXCMSG\]](#).

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.11.1 RopDeleteMessages ROP Request Buffer

The following descriptions define valid fields for the **RopDeleteMessages** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.11.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the folder that contains the messages to be soft deleted.

WantAsynchronous (1 byte): A Boolean value that is nonzero (TRUE) if the ROP is to be processed asynchronously or zero (FALSE) if the ROP is to be processed synchronously. For details about asynchronous processing, see section [3.1.5.2](#).

NotifyNonRead (1 byte): A Boolean value that is nonzero (TRUE) if a **non-read receipt** is to be generated for the deleted messages. The value is zero (FALSE) otherwise.

MessageIdCount (2 bytes): An integer that specifies the number of structures contained in the **MessageIds** field.

MessageIds (variable): An array of **MID** structures ([\[MS-OXCDATA\]](#) section 2.2.1.2), each of which specifies a message to be deleted. The number of structures contained in the array is specified by the value of the **MessageIdCount** field.

2.2.1.11.2 RopDeleteMessages ROP Response Buffer

The following descriptions define valid fields for the **RopDeleteMessages** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.11.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCDATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE). Otherwise, the value is zero (FALSE).

2.2.1.12 RopHardDeleteMessages ROP

The **RopHardDeleteMessages** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.12) is used to hard delete one or more messages from a folder.

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.12.1 RopHardDeleteMessages ROP Request Buffer

The following descriptions define valid fields for the **RopHardDeleteMessages** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.12.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the folder that contains the messages to be hard deleted.

WantAsynchronous (1 byte): A Boolean value that is nonzero (TRUE) if the ROP is to be processed asynchronously or zero (FALSE) if the ROP is to be processed synchronously. For details about asynchronous processing, see section [3.1.5.2](#).

NotifyNonRead (1 byte): A Boolean value that is nonzero (TRUE) if a non-read receipt is to be generated for the deleted messages. The value is zero (FALSE) otherwise.

MessageIdCount (2 bytes): An integer that specifies the number of structures contained in the **MessageIds** field.

MessageIds (variable): An array of **MID** structures ([\[MS-OXCDATA\]](#) section 2.2.1.2), each of which specifies a message to be deleted. The number of structures contained in the array is specified by the value of the **MessageIdCount** field.

2.2.1.12.2 RopHardDeleteMessages ROP Response Buffer

The following descriptions define valid fields for the **RopHardDeleteMessages** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.12.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCDATA\]](#) section 2.4.

PartialCompletion (1 byte): A Boolean value that specifies whether the ROP fails for a subset of targets. If the ROP fails for a subset of targets, the value of this field is nonzero (TRUE). Otherwise, the value is zero (FALSE).

2.2.1.13 RopGetHierarchyTable ROP

The **RopGetHierarchyTable** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.13) is used to retrieve the **hierarchy table** for a folder. This ROP returns a Table object on which table operations can be performed. For details about Table objects and table operations, see [\[MS-OXCTABL\]](#).

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.13.1 RopGetHierarchyTable ROP Request Buffer

The following descriptions define valid fields for the **RopGetHierarchyTable** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.13.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the folder whose hierarchy table will be retrieved.

OutputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the output Server object is stored. The output Server object for this operation is a Table object that represents the hierarchy table. For more details about Table objects, see [\[MS-OXCTABL\]](#).

TableFlags (1 byte): A set of bits that control how information is returned in the hierarchy table.

The valid bits for this field are listed in the following table. The client MUST NOT set any other bits.

Bit name	Value	Meaning
Depth	0x04	If this bit is set, the hierarchy table lists folders from all levels under the folder. If this bit is not set, the hierarchy table lists only the folder's immediate child folders.
DeferredErrors	0x08	The ROP response is returned immediately, possibly before the ROP execution is complete, and in this case, the values of the ReturnValue and the RowCount fields of the ROP response buffer might not be accurate.
NoNotifications	0x10	The hierarchy table notifications to the client are disabled. Table notifications are specified in [MS-OXCNOTIF] section 2.2.1.1.1.
SoftDeletes	0x20	If this bit is set, the hierarchy table lists only the soft-deleted folders. If this bit is not set, the hierarchy table lists only the existing folders.

Bit name	Value	Meaning
UseUnicode	0x40	If this bit is set, the columns that contain string data are returned in Unicode format. If this bit is not set, the string data is encoded in the code page of the Logon object.
SuppressesNotifications	0x80	The notifications generated by the client's actions on the hierarchy table are suppressed.

2.2.1.13.2 RopGetHierarchyTable ROP Response Buffer

The following descriptions define valid fields for the **RopGetHierarchyTable** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.13.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCDATA\]](#) section 2.4. If the **TableFlags** field of the ROP request buffer has the DeferredErrors bit set, the value of the **ReturnValue** field is valid only if it indicates a failure.

RowCount (4 bytes): An integer that specifies the number of rows in the hierarchy table. If the **TableFlags** field of the ROP request buffer has the DeferredErrors bit set, the value of the **RowCount** field might not be accurate.

2.2.1.14 RopGetContentsTable ROP

The **RopGetContentsTable** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.14) is used to retrieve the **contents table** for a folder. The contents table for a search folder (2) contains the same columns as the contents table for a generic folder. This ROP returns a Table object on which table operations can be performed. For details about Table objects and table operations, see [\[MS-OXCTABL\]](#).

The complete syntax of the ROP request buffer and the ROP response buffer is specified in [\[MS-OXCROPS\]](#). This section specifies the syntax and semantics of various fields that are not fully specified in [\[MS-OXCROPS\]](#).

2.2.1.14.1 RopGetContentsTable ROP Request Buffer

The following descriptions define valid fields for the **RopGetContentsTable** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.14.1).

InputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the input Server object is stored. The input Server object for this operation is a Folder object that represents the folder whose contents table will be retrieved.

OutputHandleIndex (1 byte): An integer that specifies the location in the Server object handle table where the handle for the output Server object is stored. The output Server object for this operation is a Table object that represents the contents table. For more details about Table objects, see [\[MS-OXCTABL\]](#).

TableFlags (1 byte): A set of bits that control how information is returned in the contents table.

The bits that SHOULD [<4>](#) be valid for this field are specified in the following table. The client MUST NOT set any other bits.

Bit name	Value	Meaning
Associated	0x02	If this bit is set, the contents table lists only the FAI messages. If this bit is not set, the contents table lists only the non-FAI messages. For details about FAI messages, see [MS-OXCMSG] section 1.3.2.
DeferredErrors	0x08	The ROP response is returned immediately, possibly before the ROP execution is complete, and in this case, the values of the ReturnValue and the RowCount fields of the ROP response buffer might not be accurate.
NoNotifications	0x10	The contents table notifications to the client are disabled. Table notifications are specified in [MS-OXCNOTIF] section 2.2.1.1.1.
SoftDeletes	0x20	If this bit is set, the contents table lists only the soft-deleted messages. If this bit is not set, the contents table lists only the existing messages.
UseUnicode	0x40	If this bit is set, the columns that contain string data are returned in Unicode format. If this bit is not set, the string data is encoded in the code page of the Logon object.
ConversationMembers	0x80	The contents table lists messages pertaining to a single conversation (one result row represents a single message). This bit is valid only in conjunction with the UseUnicode bit and the DeferredErrors bit; This bit is supported only on the Root folder of a mailbox and on public folders.

2.2.1.14.2 RopGetContentsTable ROP Response Buffer

The following descriptions define valid fields for the **RopGetContentsTable** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.4.14.2).

ReturnValue (4 bytes): An integer that indicates the result of the operation. The server returns 0x00000000 to indicate success. For details about common error codes, see [\[MS-OXCDATA\]](#) section 2.4. If the **TableFlags** field of the ROP request buffer has the DeferredErrors bit set, the value of the **ReturnValue** field is valid only if it indicates a failure.

RowCount (4 bytes): An integer that specifies the number of rows in the contents table. If the **TableFlags** field of the ROP request buffer has the DeferredErrors bit set, the value of the **RowCount** field might not be accurate.

2.2.2 Folder Object Properties

A Folder object can be created and modified by clients and servers. Except where noted, the constraints to which both clients and servers adhere when operating on a Folder object are defined in sections [2.2.2.1](#) through [2.2.2.2.6](#). Unless otherwise specified, a Folder object adheres to all property constraints specified in [\[MS-OXPROPS\]](#). A Folder object can also contain other properties, as specified in [\[MS-OXOSFLD\]](#) and [\[MS-OXOSRCH\]](#).

When a property is referred to as read-only, it means that clients SHOULD NOT try to change the value of this property and servers return an error and ignore any request to change the value of the property.

2.2.2.1 General Properties

The following properties exist on Folder objects as well as on other messaging objects. These properties are set by the server and are read-only to the client.

PidTagAccess ([\[MS-OXCPRPT\]](#) section 2.2.1.1)

PidTagChangeKey ([\[MS-OXCFXICS\]](#) section 2.2.1.2.7)

PidTagCreationTime ([\[MS-OXCMSG\]](#) section 2.2.2.3)

PidTagLastModificationTime ([\[MS-OXCMSG\]](#) section 2.2.2.2)

2.2.2.2 Folder Object Specific Properties

The properties that are specific to a Folder object are specified in sections [2.2.2.2.1](#) and in [2.2.2.2.2](#).

2.2.2.2.1 Read-Only Properties

2.2.2.2.1.1 PidTagContentCount Property

Type: **PtypInteger32** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagContentCount** property ([\[MS-OXPROPS\]](#) section 2.705) specifies the number of messages in a folder, as computed by the message store. The value does not include FAI entries in the folder.

2.2.2.2.1.2 PidTagContentUnreadCount Property

Type: **PtypInteger32** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagContentUnreadCount** property ([\[MS-OXPROPS\]](#) section 2.708) specifies the number of unread messages in a folder, as computed by the message store.

2.2.2.2.1.3 PidTagDeletedOn Property

Type: **PtypTime** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagDeletedOn** property ([\[MS-OXPROPS\]](#) section 2.729) specifies the time when the folder was soft deleted.

2.2.2.2.1.4 PidTagAddressBookEntryId Property

Type: **PtypBinary** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagAddressBookEntryId** property ([\[MS-OXPROPS\]](#) section 2.592) specifies the name-service **EntryID** of a directory object that refers to a public folder. This property is set only for public folders. For details about public folders, see [\[MS-OXCSTOR\]](#) section 1.3.1.

2.2.2.2.1.5 PidTagFolderId Property

Type: **PtypInteger64** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagFolderId** property ([\[MS-OXPROPS\]](#) section 2.758) contains a **FID** structure ([\[MS-OXCDATA\]](#) section 2.2.1.1) that uniquely identifies a folder.

2.2.2.2.1.6 PidTagHierarchyChangeNumber Property

Type: **PtypInteger32** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagHierarchyChangeNumber** property ([\[MS-OXPROPS\]](#) section 2.777) specifies the number of subfolders in the folder. The value of this property monotonically increases every time a subfolder is added to or deleted from the folder.

2.2.2.2.1.7 PidTagMessageSize Property

Type: **PtypInteger32** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagMessageSize** property ([\[MS-OXPROPS\]](#) section 2.852) specifies the aggregate size of messages in the folder.

2.2.2.2.1.8 PidTagMessageSizeExtended Property

Type: **PtypInteger64** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagMessageSizeExtended** property ([\[MS-OXPROPS\]](#) section 2.853) specifies the 64-bit version of the **PidTagMessageSize** property (section [2.2.2.2.1.7](#)).

2.2.2.2.1.9 PidTagSubfolders Property

Type: **PtypBoolean** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagSubfolders** property ([\[MS-OXPROPS\]](#) section 2.1085) specifies whether the folder has any subfolders. The value of this property is nonzero if the folder has subfolders; the value is zero otherwise.

2.2.2.2.2 Read/Write Properties

2.2.2.2.2.1 PidTagAttributeHidden Property

Type: **PtypBoolean** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagAttributeHidden** property ([\[MS-OXPROPS\]](#) section 2.669) specifies whether the folder is hidden. The value of this property is nonzero if the folder is hidden; the value is zero otherwise.

2.2.2.2.2.2 PidTagComment Property

Type: **PtypString** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagComment** property ([\[MS-OXPROPS\]](#) section 2.695) contains a comment about the purpose or content of the folder. This property is present only if the client sets it when the folder is created. The client sets the property by specifying a comment in the **Comment** field of the **OpCreateFolder** request (section [2.2.2.1](#)).

2.2.2.2.2.3 PidTagDisplayName Property

Type: **PtypString** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagDisplayName** property ([\[MS-OXPROPS\]](#) section 2.734) specifies the display name of the folder. Sibling folders MUST have unique display names.

2.2.2.2.2.4 PidTagFolderType Property

Type: **PtypInteger32** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagFolderType** property ([\[MS-OXPROPS\]](#) section 2.759) specifies the type of the folder.

The valid values of the **PidTagFolderType** property are listed in the following table.

Folder type	Value	Description
FOLDER_ROOT	0	The Root folder of the folder hierarchy table; that is, a folder that has no parent folder.
FOLDER_GENERIC	1	A generic folder that contains messages and other folders.
FOLDER_SEARCH	2	A folder that contains the results of a search, in the form of links to messages that meet search criteria.

2.2.2.2.2.5 PidTagRights Property

Type: **PtypInteger32** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagRights** property ([\[MS-OXPROPS\]](#) section 2.985) specifies the client's folder permissions. This property's format is the same as that of the **PidTagMemberRights** property ([\[MS-OXCPerm\]](#) section 2.2.1.7). The FreeBusyDetailed flag and the FreeBusySimple flag do not apply to the **PidTagRights** property.

2.2.2.2.2.6 PidTagAccessControlListData Property

Type: **PtypBinary** ([\[MS-OXCDATA\]](#) section 2.11.1)

The **PidTagAccessControlListData** property ([\[MS-OXCPerm\]](#) section 2.2.1.3) contains the **access control list (ACL)** of the folder. [<5>](#)

3 Protocol Details

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 specified in this document.

3.1.1.1 Hierarchy Table

A hierarchy table contains information about the folders in a message store. Each row of a hierarchy table contains a set of columns with information about one folder. Hierarchy tables are used primarily by clients and implemented by message-store providers to show a tree of folders and subfolders.

The following are the two types of hierarchy tables:

- Standard
- Soft deleted

The standard table contains only folders that were not deleted. The soft-deleted table contains only folders that have been soft deleted.

A hierarchy table can be obtained by using the **RopGetHierarchyTable** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.13).

3.1.1.2 Contents Table

A contents table contains information about objects in a message search folder container. The contents table of a folder lists information about its messages.

The following are the four types of contents tables:

- Standard
- Standard soft deleted
- FAI
- FAI soft deleted

Standard contents tables contain only standard (non-FAI) messages. FAI tables contain only FAI messages. For more information about FAI messages, see [\[MS-OXCMSG\]](#) section 1.3.2.

The soft-deleted views contain only messages that have been soft deleted.

A contents table is obtained by using the **RopGetContentsTable** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.14).

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Opening a Folder

Before any data can be read from or written to a folder, the client opens the folder. The client MUST have sufficient access rights to the folder for this operation to succeed.

To open an existing folder, the client sends the **RopOpenFolder** ROP request ([MS-OXCROPS] section 2.2.4.1). To send this request, the client first obtains the FID ([MS-OXCDATA] section 2.2.1.1) for the Folder object to be opened. The FID can be retrieved from the hierarchy table that contains the folder's information by including the **PidTagFolderId** property (section 2.2.2.1.5) in a **RopSetColumns** request ([MS-OXCROPS] section 2.2.5.1.1).

The Folder object that is opened can be used in subsequent operations. When the client completes operations on the Folder object, the client MUST release the object as specified in section 3.1.5.1.

3.1.4.2 Creating a Folder

Before any data can be read from or written to a folder, the client opens the folder as specified in section 3.1.4.1. If the folder does not exist, the client creates the folder. Before a folder can be created, the parent folder MUST already exist.

To create a folder, the client sends the **RopCreateFolder** ROP request ([MS-OXCROPS] section 2.2.4.2).

The Folder object that is returned by the **RopCreateFolder** ROP can be used in subsequent operations. When the client completes operations on the Folder object, the client MUST release the object as specified in section 3.1.5.1.

3.1.4.3 Deleting a Folder

To be deleted, a folder MUST exist, and the client needs the access rights to delete it. To delete a folder, the client sends a **RopDeleteFolder** ROP request ([MS-OXCROPS] section 2.2.4.3). If the folder is not empty, the client sets the **DeleteFolderFlags** field to delete all existing subfolders and messages, as specified in section 2.2.1.3.1. The **DeleteFolderFlags** field can also be used to specify a hard deletion, when the DELETE_HARD_DELETE bit is set. The **PartialCompletion** field of the ROP response, as specified in section 2.2.1.3.2, indicates whether there are any subfolders or messages that could not be deleted and, consequently, that the folder was not deleted.

3.1.4.4 Setting Up a Search Folder

The client creates a search folder (2) by using the **RopCreateFolder** ROP ([MS-OXCROPS] section 2.2.4.2) with the **FolderType** field set to the value 2, as specified in section 2.2.1.2.1. For details about creating a folder, see section 3.1.4.2.

The client fills a search folder (2) by applying search criteria and a search scope to the search folder (2). The search criteria determine which messages are included in the folder, and the search scope determines the folders that are searched.

To set the search criteria and search scope for a search folder (2), the client sends a **RopSetSearchCriteria** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.4) with bits of the **SearchFlags** field set to control the details of how the search is performed, as specified in section [2.2.1.4.1](#). The ROP request includes restrictions (2) to specify the search criteria and FIDs ([\[MS-OXCADATA\]](#) section 2.2.1.1) to specify the search scope.

To access the results of a search, the client sends a **RopGetContentsTable** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.14) to obtain the contents table of the search folder (2), as specified in section [3.1.4.10](#). The messages that match the search criteria are specified in the table.

When the client is finished using a search folder (2), the folder can either be deleted or remain open for later use. Note that if the search folder (2) is deleted, only message links are deleted. The actual messages remain in their parent folders.

3.1.4.5 Getting Details About a Search Folder

To obtain details about a search folder (2), the client sends a **RopGetSearchCriteria** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.5) with the appropriate values in the **IncludeRestriction** and **IncludeFolders** fields, as specified in section [2.2.1.5.1](#), to specify the details to be included in the response. The details about a search folder (2) can include the search criteria, a list of the folders that are being searched, and the status of the search folder (2).

For details about how the client sets up a search folder (2), see section [3.1.4.4](#).

3.1.4.6 Moving a Folder and Its Contents

To move particular messages from one folder to another, the client sends a **RopMoveCopyMessages** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.6) with the **WantCopy** field set to zero (FALSE), as specified in section [2.2.1.6.1](#). The client's request includes a list of MIDs ([\[MS-OXCADATA\]](#) section 2.2.1.2) that specify the messages to be moved.

To move a folder from one parent folder to another, the client sends a **RopMoveFolder** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.7). All of the folder's messages and subfolders are moved with the folder.

The **RopMoveCopyMessages** ROP request and **RopMoveFolder** ROP request are processed asynchronously if the client sets the **WantAsynchronous** field of the request to nonzero (TRUE), as specified in sections [2.2.1.6.1](#) and [2.2.1.7.1](#). For details about asynchronous processing, see section [3.1.5.2](#).

3.1.4.7 Copying a Folder and Its Contents

To copy particular messages from one folder to another, the client sends a **RopMoveCopyMessages** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.6) with the **WantCopy** field set to nonzero (TRUE), as specified in section [2.2.1.6.1](#). The client's request includes a list of MIDs ([\[MS-OXCADATA\]](#) section 2.2.1.2) that specify the messages to be copied.

To copy a folder from one parent folder to another, the client sends a **RopCopyFolder** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.8). All of the messages in the source folder are duplicated in the new folder. If the **WantRecursive** field is set to nonzero (TRUE), as specified in section [2.2.1.8.1](#), the subfolders that are contained in the source folder are also duplicated in the new folder.

The **RopMoveCopyMessages** ROP request and **RopCopyFolder** ROP request are processed asynchronously if the client sets the **WantAsynchronous** field of the request to nonzero (TRUE), as specified in sections [2.2.1.6.1](#) and [2.2.1.7.1](#). For details about asynchronous processing, see section [3.1.5.2](#).

3.1.4.8 Deleting the Contents of a Folder

To delete all messages and subfolders from a folder without deleting the folder itself, the client sends either a **RopEmptyFolder** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.9) or a **RopHardDeleteMessagesAndSubfolders** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.10). The client uses the **RopEmptyFolder** ROP for a soft delete and the **RopHardDeleteMessagesAndSubfolders** ROP for a hard delete. A message that has been soft deleted can be recovered only by opening it, copying its contents to a new message, and then saving the new message. For details about opening, copying, and saving a Message object, see [\[MS-OXCMSG\]](#).

To remove particular messages from a folder, the client sends either a **RopDeleteMessages** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.11) or a **RopHardDeleteMessages** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.12). The client uses **RopDeleteMessages** for a soft delete and **RopHardDeleteMessages** for a hard delete.

The **RopEmptyFolder** ROP request, **RopHardDeleteMessagesAndSubfolders** ROP request, **RopDeleteMessages** ROP request, and **RopHardDeleteMessages** ROP request are processed asynchronously if the client sets the **WantAsynchronous** field of the request to nonzero (TRUE), as specified in sections [2.2.1.9.1](#), [2.2.1.10.1](#), [2.2.1.11.1](#), and [2.2.1.12.1](#). For details about asynchronous processing, see section [3.1.5.2](#).

3.1.4.9 Getting a Hierarchy Table

To retrieve a hierarchy table that is associated with a folder, the client sends a **RopGetHierarchyTable** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.13) with the appropriate bits set in the **TableFlags** field, as specified in section [2.2.1.13.1](#).

Subsequent operations can be executed on the Table object that is returned by the **RopGetHierarchyTable** ROP. When the client completes operations on the Table object, the client MUST release the object as specified in section [3.1.5.1](#).

3.1.4.10 Getting a Contents Table

To retrieve a contents table that is associated with a folder, the client sends a **RopGetContentsTable** ROP request ([\[MS-OXCROPS\]](#) section 2.2.4.14) with the appropriate bits set in the **TableFlags** field, as specified in section [2.2.1.14.1](#).

For conversation view, the client sets the ConversationMembers bit in the **TableFlags** field of the **RopGetContentsTable** request. After sending this request, the client sends a **RopRestrict** request ([\[MS-OXCROPS\]](#) section 2.2.5.3) to limit the view to the messages of a particular conversation, as specified in [\[MS-OXCTABL\]](#).

Subsequent operations can be executed on the Table object that is returned by **RopGetContentsTable** ROP. When the client completes operations on the Table object, the client MUST release the object as specified in section [3.1.5.1](#).

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Releasing a Server Object

The client MUST send a **RopRelease** ROP request ([MS-OXCROPS] section 2.2.15.3.1) to release the Server object after executing all subsequent operations on the folder that was created or opened. Likewise, the client MUST send a **RopRelease** ROP request after executing all subsequent operations on a contents table or a hierarchy table.

If the client does not send a **RopRelease** ROP request, the server does not free the resources associated with the Folder object or the Table object. For details about Server object dependencies and releasing resources, see [MS-OXCROPS] sections [3.1.5.3](#) and [3.2.5.3](#).

3.1.5.2 Processing ROPs Asynchronously

Some ROPs can be processed asynchronously. During asynchronous processing, the client receives a **RopProgress** ROP response buffer ([MS-OXCROPS] section 2.2.8.13) instead of the ROP response buffer that corresponds to the ROP request buffer that the client sent. The **RopProgress** ROP response indicates that the operation is still being processed. When processing is complete, the client receives the ROP response buffer that corresponds to the original ROP request buffer.

Any of the following ROPs can be processed asynchronously if its ROP request buffer was sent with the **WantAsynchronous** field set to nonzero (TRUE).

RopMoveCopyMessages (section [2.2.1.6](#))

RopMoveFolder (section [2.2.1.7](#))

RopCopyFolder (section [2.2.1.8](#))

RopEmptyFolder (section [2.2.1.9](#))

RopHardDeleteMessagesAndSubfolders (section [2.2.1.10](#))

RopDeleteMessages (section [2.2.1.11](#))

RopHardDeleteMessages (section [2.2.1.12](#))

The client can send a **RopProgress** ROP request buffer either to abort an in-progress operation or to get information about the progress of an operation. For more details about the **RopProgress** ROP, see [MS-OXCPRPT] sections [2.2.22](#) and [3.2.5.19](#).

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 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 specified in this document.

The abstract data model used by the server and the client are the same.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

Various agents on the server could issue the same higher-layer triggered events, as specified in section 3.1.4. The same considerations specified in section 3.1.4 for client implementations also apply to server implementations.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Processing a RopOpenFolder ROP Request

When the server receives a **RopOpenFolder** ROP request buffer ([MS-OXCROPS] section 2.2.4.1) from the client, the server parses the buffer. The server responds with a **RopOpenFolder** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [MS-OXCROPS] section 3.2.5.1. For details about how the server formats buffers for the response, see [MS-OXCROPS] section 3.2.5.2.

The **RopOpenFolder** ROP succeeds only if a folder with the specified ID actually exists <6> and the client has sufficient access rights to view the folder.

If the OpenSoftDeleted bit of the **OpenModeFlags** field of the ROP request buffer is set, as specified in section 2.2.1.1.1, the server provides access to folders that are soft deleted and to folders that are not soft deleted. If the bit is not set, the server provides access only to folders that are not soft deleted. The server MUST ignore any invalid bits that are set in the **OpenModeFlags** field.

The following specific error codes apply to this ROP. For more details about ROP errors, see [MS-OXCADATA] section 2.4.

Error code name	Value	Meaning
ecNotFound	0x8004010F	The FID ([MS-OXCADATA] section 2.2.1.1) does not correspond to a folder in the database, or the client does not have rights to the folder, or the folder is soft-deleted and the client has not specified the OpenSoftDeleted bit in the OpenModeFlags field.
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object or Logon object.

3.2.5.2 Processing a RopCreateFolder ROP Request

When the server receives a **RopCreateFolder** ROP request buffer ([MS-OXCROPS] section 2.2.4.2) from the client, the server parses the buffer. The server responds with a **RopCreateFolder** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [MS-OXCROPS] section 3.2.5.1. For details about how the server formats buffers for the response, see [MS-OXCROPS] section 3.2.5.2.

A folder name in the **DisplayName** field of the ROP request buffer, as specified in section 2.2.1.2.1, MUST be specified to create a folder. A folder description, specified in the **Comment** field of the ROP request buffer, is optional. The folder name MUST be unique within the parent folder. In other words, sibling folders cannot have the same name.

If a folder with the same name already exists, and the **OpenExisting** field is set to zero (FALSE), the server fails the **RopCreateFolder** ROP request with error code ecDuplicateName. If a folder with the same name already exists and the **OpenExisting** field is set to nonzero (TRUE), the server opens the existing folder, behaving as if it is processing the **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1). If a folder with the same name does not exist, the server creates a new folder, regardless of the value of the **OpenExisting** field.

If the client does not have permissions to create the folder, the server returns either ecAccessdenied or ecNoCreateSubfolderRight. <7> If the client attempts to create a search folder (2) on a public folders store, the server returns either ecNotSupported or ecError. <8>

The following specific error codes apply to this ROP. For more details about ROP errors, see [MS-OXCROPS] section 2.4.

Error code name	Value	Meaning
ecInvalidParam	0x80070057	The FolderType field contains an invalid value.
ecError	0x80004005	The operation failed for an unspecified reason.
ecAccessdenied	0x80070005	The client does not have permissions to create the folder.
ecDuplicateName	0x80040604	A folder with the same name already exists, and the OpenExisting field was set to zero (FALSE).
ecNotSupported	0x80040102	The ROP was called on an object that is not a Folder object, or the client attempted to create a search folder (2) on a public folders store.
ecNotFound	0x8004010F	The ROP was called on a Folder object that is a soft-deleted folder.
ecNoCreateSubfolderRight	0x00000502	The client does not have access rights to create the folder.

3.2.5.3 Processing a RopDeleteFolder ROP Request

When the server receives a **RopDeleteFolder** ROP request buffer ([MS-OXCROPS] section 2.2.4.3) from the client, the server parses the buffer. The server responds with a **RopDeleteFolder** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [MS-OXCROPS] section 3.2.5.1. For details about how the server formats buffers for the response, see [MS-OXCROPS] section 3.2.5.2.

If the DELETE_HARD_DELETE bit of the **DeleteFolderFlags** field of the ROP request buffer is set, as specified in section 2.2.1.3.1, the folder MUST be removed and can no longer be accessed by the

client with subsequent ROPs. If the DELETE_HARD_DELETE bit is not set, the folder becomes soft deleted.

If the DEL_MESSAGES bit is not set and the folder contains Message objects, neither the folder nor any of its Message objects will be deleted. If the DEL_FOLDERS bit is not set and the folder contains subfolders, neither the folder nor any of its subfolders will be deleted. In both cases, the **ReturnValue** field of the ROP response, as specified in section 2.2.1.3.2, will be set to 0x00000000 and the **PartialCompletion** field will be set to a nonzero (TRUE) value.

The server MUST ignore any invalid bits that are set in the **DeleteFolderFlags** field of the ROP request buffer.

The following specific error codes apply to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecAccessDenied	0x80070005	An attempt was made to delete a special folder , or the client does not have permissions to delete this folder.
ecNotFound	0x8004010F	Folder with the specified ID does not exist, or the client has no access to view that folder.
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object, or an attempt was made to delete the Root folder.

3.2.5.4 Processing a RopSetSearchCriteria ROP Request

When the server receives a **RopSetSearchCriteria** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.4) from the client, the server parses the buffer. The server responds with a **RopSetSearchCriteria** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server fills the search folder (2) according to the search criteria and search scope that are specified in the **RopSetSearchCriteria** ROP request. The messages that satisfy the search criteria appear as links in the search folder (2). When new search criteria are applied, the server modifies the search folder (2) to include only the messages that match the new search criteria. The server can return the **RopSetSearchCriteria** ROP response before the search folder (2) is fully updated.

For static search folders (2), the contents of the search folder (2) are not updated after the initial population is complete. For dynamic search folders (2), the contents of the search folder (2) MUST continue to be updated as messages move around the mailbox and start to match or cease to match the search criteria. A static search causes the search folder (2) to be populated once with all messages that match the search criteria at the point in time when the search is started or restarted. The server MUST NOT update the search folder (2) after the initial population when new messages that match the search criteria arrive in the search scope or when existing messages that fit the search criteria are deleted. To trigger an update, another **RopSetSearchCriteria** ROP request with the RESTART_SEARCH bit set in the **SearchFlags** field, as specified in section 2.2.1.4.1, is required. A dynamic search causes the search folder (2) to be initially populated with all messages that match the search criteria at the point in time when the search is started or restarted. The server continues to update the search folder (2) with messages that enter or exit the search criteria. A **RopSetSearchCriteria** ROP request with the STOP_SEARCH bit set does not have any effect on a dynamic search that has already completed its initial population and also does not change the dynamic nature of the search folder (2).

The server can use context indexing by default at the discretion of the server implementation. Whether it is used is usually based on the nature of the restriction (2) that is used. When using context indexing in searches, the server allows the client to quickly search text in messages through the use of prebuilt indexes, while searches that are not content-indexed are based on a sequential scan of all the messages in the search scope.

Some differences between searches that are content-indexed and those that are not content-indexed are listed in the following table.

Content-indexed	Not content-indexed
Based on words, phrases, and sentences.	Based on a stream of bytes.
Ignores punctuation and spaces and is also not case sensitive.	Finds only an exact match of all characters.
Searches within attachment types that are supported by the installed filters.	Does not search within attachments.
Uses full-text index to locate records.	Performs a serial scan of the entire folder.
Supports only full-text searches .	Supports the full set of restrictions (2), which includes nontext property types such as date and time.

If the NON_CONTENT_INDEXED_SEARCH bit is set in the **SearchFlags** field, the search does not use a full-text search. If the NON_CONTENT_INDEXED_SEARCH bit is not set, the server uses a content-indexed search for text searches, in which case the search is static, regardless of the STATIC_SEARCH bit.

If the STATIC_SEARCH bit is set in the **SearchFlags** field, the search is static. If the NON_CONTENT_INDEXED_SEARCH bit is set and the STATIC_SEARCH bit is not set, the search is a dynamic search, not a full-text search.

If the STOP_SEARCH bit is set in the **SearchFlags** field, the server SHOULD stop the initial population of the search folder (2). Due to the asynchronous nature of the call, the server can complete the operation before the **RopSetSearchCriteria** ROP request with STOP_SEARCH is serviced. The server can take some time to stop and might not stop at all. If the RESTART_SEARCH bit is set in the **SearchFlags** field, the server restarts the population of the search folder (2). If neither the STOP_SEARCH bit nor RESTART_SEARCH bit is set, the search continues in the previous state and either continues populating or not.

If the client sets an invalid bit in the **SearchFlags** field, the server SHOULD [fail](#) the ROP with the error code ecInvalidParam (0x80070057).

The following specific error codes apply to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecInvalidParam	0x80070057	The SearchFlags field contains an invalid value.
ecNotInitialized	0x80040605	No FIDs ([MS-OXCDATA] section 2.2.1.1) have been specified for this search folder (2). Note that if the FIDs were specified in a previous RopSetSearchCriteria ROP request and no IDs are specified in the next RopSetSearchCriteria ROP request, the previous IDs will continue to be used. If FIDs are specified, they will override previous IDs.

Error code name	Value	Meaning
ecNotSearchFolder	0x00000461	The object is not a search folder (2).
ecTooComplex	0x80040117	The restriction (2) is too complex. The definition of complexity is server-specific.
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object, or the request specified a recursive search on a public folder.

3.2.5.5 Processing a RopGetSearchCriteria ROP Request

When the server receives a **RopGetSearchCriteria** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.5) from the client, the server parses the buffer. The server responds with a **RopGetSearchCriteria** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server returns the search criteria only if the **IncludeRestriction** field of the ROP request buffer is set to nonzero (TRUE), as specified in section [2.2.1.5.1](#). The server returns a list of the folders that are being searched only if the **IncludeFolders** field of the ROP request buffer is set to nonzero (TRUE).

The following specific error codes apply to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSearchFolder	0x00000461	The object is not a search folder (2).
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object.

3.2.5.6 Processing a RopMoveCopyMessages ROP Request

When the server receives a **RopMoveCopyMessages** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.6) from the client, the server parses the buffer. The server responds with a **RopMoveCopyMessages** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

If the server fails to move or copy any message, it sets the **PartialCompletion** field of the **RopMoveCopyMessages** ROP response buffer to nonzero (TRUE), as specified in section [2.2.1.6.2](#).

If the client requests asynchronous execution, the server executes this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopMoveCopyMessages** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **RopMoveCopyMessages** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see [\[MS-OXCPRPT\]](#) sections [2.2.22](#) and [3.2.5.19](#).

The following specific error code applies to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102	Either the source object or the destination object is not a Folder object, or either the source object or the destination object is a search folder (2).

3.2.5.7 Processing a RopMoveFolder ROP Request

When the server receives a **RopMoveFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.7) from the client, the server parses the buffer. The server responds with a **RopMoveFolder** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

If the server fails to move any folder, message, or subfolder, it sets the **PartialCompletion** field of the **RopMoveFolder** ROP response buffer to nonzero (TRUE), as specified in section [2.2.1.7.2](#).

If the client requests asynchronous execution, the server executes this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopMoveFolder** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **RopMoveFolder** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see [\[MS-OXCPRPT\]](#) sections [2.2.22](#) and [3.2.5.19](#).

The following specific error codes apply to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotFound	0x8004010F	There is no folder with the specified ID.
ecNotSupported	0x80040102	Either the source object or the destination object is not a Folder object, or either the source object or the destination object is a search folder (2).

3.2.5.8 Processing a RopCopyFolder ROP Request

When the server receives a **RopCopyFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.8) from the client, the server parses the buffer. The server responds with a **RopCopyFolder** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

All messages contained in the source folder MUST be duplicated in the new folder. If the server fails to copy any folder, message, or subfolder, it sets the **PartialCompletion** field of the **RopCopyFolder** ROP response buffer to nonzero (TRUE), as specified in section [2.2.1.8.2](#). If the **WantRecursive** field of the **RopCopyFolder** ROP request buffer is set to nonzero (TRUE), as specified in section [2.2.1.8.1](#), the subfolders contained in the source folder are also duplicated in the new folder in a recursive manner.

If the client requests asynchronous execution, the server executes this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([\[MS-](#)

[OXCROPS](#) section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopCopyFolder** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **RopCopyFolder** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see [\[MS-OXCPRPT\]](#) sections [2.2.22](#) and [3.2.5.19](#).

The following specific error codes apply to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotFound	0x8004010F	There is no folder with the specified ID.
ecNotSupported	0x80040102	Either the source object or the destination object is not a Folder object, or either the source object or the destination object is a search folder (2).

3.2.5.9 Processing a RopEmptyFolder ROP Request

When the server receives a **RopEmptyFolder** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.9) from the client, the server parses the buffer. The server responds with a **RopEmptyFolder** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server soft deletes the folder's messages and subfolders but does not delete the folder itself. For any folder other than the **Drafts folder**, the server MUST NOT delete messages that are open with read/write access. If the server fails to delete any message or subfolder, it sets the **PartialCompletion** field of the **RopEmptyFolder** ROP response buffer to nonzero (TRUE), as specified in section [2.2.1.9.2](#).

If the **WantDeleteAssociated** field of the **RopEmptyFolder** ROP request buffer is set to nonzero (TRUE), as specified in section [2.2.1.9.1](#), then the server removes all FAI messages in addition to the **normal messages**. The server removes all subfolders regardless of the value of the **WantDeleteAssociated** field.

If the client requests asynchronous execution, the server executes this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopEmptyFolder** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **RopEmptyFolder** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see [\[MS-OXCPRPT\]](#) sections [2.2.22](#) and [3.2.5.19](#).

The following specific error code applies to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object.

3.2.5.10 Processing a RopHardDeleteMessagesAndSubfolders ROP Request

When the server receives a **RopHardDeleteMessagesAndSubfolders** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.10) from the client, the server parses the buffer. The server responds with a **RopHardDeleteMessagesAndSubfolders** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server's behavior is the same as that specified for the **RopEmptyFolder** ROP in section [3.2.5.9](#), except that messages and subfolders are hard deleted instead of soft deleted.

If the client requests asynchronous execution, the server executes this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopHardDeleteMessagesAndSubfolders** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **RopHardDeleteMessagesAndSubfolders** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see [\[MS-OXCPRPT\]](#) sections [2.2.22](#) and [3.2.5.19](#).

The following specific error code applies to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object.

3.2.5.11 Processing a RopDeleteMessages ROP Request

When the server receives a **RopDeleteMessages** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.11) from the client, the server parses the buffer. The server responds with a **RopDeleteMessages** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server soft deletes the specified messages. The server MUST NOT delete any messages that are opened with read/write access. If the server fails to delete any messages, it sets the **PartialCompletion** field of the **RopDeleteMessages** ROP response buffer to nonzero (TRUE), as specified in section [2.2.1.11.2](#).

If the **NotifyNonRead** field of the **RopDeleteMessages** ROP request buffer is set to nonzero (TRUE), as specified in section [2.2.1.11.1](#), the server generates a non-read receipt for each message that is being deleted and has requested a **read receipt**. For more information about read receipts and non-read receipts, see [\[MS-OXOMSG\]](#).

If the client requests asynchronous execution, the server executes this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopDeleteMessages** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **RopDeleteMessages** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see [\[MS-OXCPRPT\]](#) sections [2.2.22](#) and [3.2.5.19](#).

The following specific error code applies to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object.

3.2.5.12 Processing a RopHardDeleteMessages ROP Request

When the server receives a **RopHardDeleteMessages** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.12) from the client, the server parses the buffer. The server responds with a **RopHardDeleteMessages** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server's behavior is the same as that specified for the **RopDeleteMessages** ROP in section [3.2.5.11](#), except that messages are hard deleted instead of soft deleted.

If the client requests asynchronous execution, the server executes this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([\[MS-OXCROPS\]](#) section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopHardDeleteMessages** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **RopHardDeleteMessages** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see [\[MS-OXCPRPT\]](#) sections [2.2.22](#) and [3.2.5.19](#).

The following specific error code applies to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object.

3.2.5.13 Processing a RopGetHierarchyTable ROP Request

When the server receives a **RopGetHierarchyTable** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.13) from the client, the server parses the buffer. The server responds with a **RopGetHierarchyTable** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server returns a hierarchy table on which table operations can be performed. For details about Table objects and table operations, see [\[MS-OXCTABL\]](#). The Table object that is returned MUST allow access to the subfolders of the Folder object on which the **RopGetHierarchyTable** ROP is executed.

The particular information that the server returns in the hierarchy table is determined by the bit settings in the **TableFlags** field of the **RopGetHierarchyTable** ROP request buffer, as specified in section [2.2.1.13.1](#). If the DeferredErrors bit of the **TableFlags** field is set, the server can return, at its discretion, the ROP response buffer immediately, possibly before the ROP execution is complete. If the server has not completed execution of the ROP, the values of the **ReturnValue** and the **RowCount** fields of the ROP response buffer, as specified in section [2.2.1.13.2](#), might not be accurate.

If the client sets an invalid bit in the **TableFlags** field, the server SHOULD [<10>](#) fail the ROP with an error code of ecNotSupported (0x80040102).

The following specific error code applies to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object.

3.2.5.14 Processing a RopGetContentsTable ROP Request

When the server receives a **RopGetContentsTable** ROP request buffer ([\[MS-OXCROPS\]](#) section 2.2.4.14) from the client, the server parses the buffer. The server responds with a **RopGetContentsTable** ROP response buffer. For details about how the server parses buffers and processes ROPs, see [\[MS-OXCROPS\]](#) section 3.2.5.1. For details about how the server formats buffers for the response, see [\[MS-OXCROPS\]](#) section 3.2.5.2.

The server returns a contents table on which table operations can be performed. For details about Table objects and table operations, see [\[MS-OXCTABL\]](#). The Table object that is returned provides information about messages that are directly under the Folder object on which this ROP is executed. The contents table for a search folder (2) contains the same columns as the contents table for a generic folder. However, for a search folder (2), the **PidTagParentEntryId** property ([\[MS-OXPROPS\]](#) section 2.908) contains the **entry ID** of the folder where the linked message resides.

The particular information that the server returns in the contents table is determined by the bit settings in the **TableFlags** field of the **RopGetContentsTable** ROP request buffer, as specified in section [2.2.1.14.1](#). If the ConversationMembers bit of the **TableFlags** field is set, the server SHOULD [<11>](#) return a contents table that shows messages pertaining to a particular conversation. If the DeferredErrors bit of the **TableFlags** field is set, the server can return, at its discretion, the ROP response buffer immediately, possibly before the ROP execution is complete. If the server has not completed execution of the ROP, the values of the **ReturnValue** and the **RowCount** fields of the ROP response buffer, as specified in section [2.2.1.14.2](#), might not be accurate.

If the ConversationMembers bit of the **TableFlags** field is set and any bits other than the UseUnicode bit and the DeferredErrors bit are also set, the server SHOULD fail the operation with an error code of ecInvalidParam (0x80070057). If the client sets an invalid bit in the **TableFlags** field, the server SHOULD [<12>](#) fail the ROP with an error code of ecInvalidParam.

The following specific error codes apply to this ROP. For more details about ROP errors, see [\[MS-OXCDATA\]](#) section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102	The object that this ROP was called on is not a Folder object.
ecInvalidParam	0x80070057	An invalid value was specified in a field.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

The following examples illustrate the byte order of ROPs in a buffer being prepared for transmission. Note that the examples in this section show only the relevant portions of the specified ROPs; this is not the final byte sequence that gets transmitted over the wire. Also note that the data format for a multibyte field appears in **little-endian** format, with the bytes in the field presented from least significant to most significant.

Frequently, these ROP requests are packed with other ROP requests, compressed, obfuscated, and then packed in one or more **remote procedure calls (RPCs)**. These examples assume that the client has already successfully logged on to the server and has obtained any **Server object handles** that are to be used as inputs for the ROPs. For information about RPCs, see [\[MS-OXCRPC\]](#).

Examples in this section use the following format for byte sequences, expressed in hexadecimal:

```
0080: 45 4D 53 4D 44 42 2E 44-4C 4C 00 00 00 00 00 00
```

The value at the far left (0080) is the byte sequence's offset from the beginning of the buffer. Following the offset is a series of up to 16 bytes, with each two-character sequence describing the value of one byte. Here, the first byte (45) in the series is located 0x80 bytes (128 bytes) from the beginning of the buffer. The seventh byte (2E) in the series is located 0x86 bytes (134 bytes) from the beginning of the buffer. The dash between the eighth byte (44) and the ninth byte (4C) has no semantic value and serves only to distinguish the eight-byte boundary for readability.

This byte sequence is followed by one or more lines that interpret it. In larger examples, the byte sequence is shown once in its entirety and then repeated in smaller chunks, with each smaller chunk interpreted separately.

When explaining **InputHandleIndex** values, the example text describes the Server object that is referenced by the handle index. For information about Server object handles, see [\[MS-OXCROPS\]](#) section 1.3.1.

4.1 Creating a New Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopCreateFolder** operation, as described in [\[MS-OXCROPS\]](#) section 2.2.4.2.

4.1.1 Client Request Buffer

The client request buffer for the **RopCreateFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.2) example consists of a 26-byte sequence, formatted as follows.

```
0000: 1C 00 00 01 01 01 00 00-46 00 6F 00 6C 00 64 00
0010: 65 00 72 00 31 00 00 00-00 00
```

The first four bytes refer to the **RopId** and **LogonId** fields, as described in [\[MS-OXCROPS\]](#) section 2.2.4.2.1, and the **InputHandleIndex** and **OutputHandleIndex** fields of the **RopCreateFolder** ROP format, as described in section [2.2.1.2.1](#).

```
0000: 1C 00 00 01
```

RopId: 0x1C (**RopCreateFolder** ROP)

LogonId: 0x00

InputHandleIndex: 0x00. The location where the handle for the input folder is stored.

OutputHandleIndex: 0x01. The the location where the handle for the newly created folder is stored.

The next four bytes contain the **FolderType**, **UseUnicodeStrings**, **OpenExisting**, and **Reserved** fields of the **RopCreateFolder** format, as specified in section [2.2.1.2.1](#). These fields affect how the operation is carried out.

```
0004: 01 01 00 00
```

FolderType: 0x01 (generic). The folder is a generic folder.

UseUnicodeStrings: 0x01 (TRUE). The folder name is in Unicode format.

OpenExisting: 0x00 (FALSE). The operation will fail if the folder already exists.

Reserved: 0x00 (FALSE).

The next 16 bytes contain the **DisplayName** field, as described in section [2.2.1.2.1](#). This field is formatted as Unicode text, as indicated by the value sent in the **UseUnicodeStrings** field.

```
0008: 46 00 6F 00 6C 00 64 00-65 00 72 00 31 00 00 00
```

DisplayName: "Folder1"

The **Comment** field, as specified in section [2.2.1.2.1](#), is sent next and, in this example, is a null-terminated string that consists of zero (0) characters and follows the same text format (Unicode) as the **DisplayName** field.

```
0018: 00 00
```

Comment: ""

4.1.2 Server Responds to Client Request

The server response buffer for the successful **RopCreateFolder** operation ([\[MS-OXCROPS\]](#) section 2.2.4.2) consists of a 15-byte sequence, formatted as follows.

```
0000: 1C 01 00 00 00 00 01 00-00 00 0E 91 52 12 00
```

The first six bytes contain the **RopId** and **OutputHandleIndex** fields, as specified in [\[MS-OXCROPS\]](#) section 2.2.4.2.2, and the **ReturnValue** field, as specified in section [2.2.1.2.2](#).

```
0000: 1C 01 00 00 00 00
```

RopId: 0x1C (**RopCreateFolder** ROP)

OutputHandleIndex: 0x01. This index is the same as that in the **OutputHandleIndex** field specified in the request in section [4.1.1](#).

ReturnValue: 0x00000000. The folder has successfully been created.

The next eight bytes provide the **FolderId** field, as described in section [2.2.1.2.2](#), for the newly created folder.

```
0006: 01 00 00 00 0E 91 52 12
```

FolderId: 0001-00000E915212

The next byte contains the **IsExistingFolder** field, as described in section [2.2.1.2.2](#).

```
000F: 00
```

IsExistingFolder: 0x00 (FALSE). A new folder was created.

Because the value of the **IsExistingFolder** field is FALSE, this is the last byte of this ROP response buffer.

4.2 Deleting an Existing Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopDeleteFolder** operation, as described in [\[MS-OXCROPS\]](#) section 2.2.4.3.

4.2.1 Client Request Buffer

The client request buffer for the **RopDeleteFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.3) example consists of a 12-byte sequence, formatted as follows.

```
0000: 1D 00 01 05 01 00 00 00-0E 8E DF 36
```

RopId: 0x1D (**RopDeleteFolder** ROP)

LogonID: 0x00

InputHandleIndex: 0x01. The location where the handle for the folder is stored.

DeleteFolderFlags: 0x05 (DEL_MESSAGES | DEL_FOLDERS). The specified folder and all messages and subfolders within the folder have to be deleted.

FolderId: 0001-00000E8EDF36. This field uniquely identifies the folder to be deleted.

4.2.2 Server Responds to Client Request

The server response buffer for the successful **RopDeleteFolder** operation ([\[MS-OXCROPS\]](#) section 2.2.4.3) consists of a 7-byte sequence, formatted as follows.

```
0000: 1D 01 00 00 00 00 00
```

RopId: 0x1D (**RopDeleteFolder** ROP)

InputHandleIndex: 0x01. This index is the same as that in the **InputHandleIndex** field in the request in section [4.2.1](#).

ReturnValue: 0x00000000. The folder has been deleted.

PartialCompletion: 0x00 (FALSE). The operation was completed and all messages and folders specified in the ROP request were deleted.

4.3 Deleting Messages Within a Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopDeleteMessages** operation, as described in [\[MS-OXCROPS\]](#) section 2.2.4.11. In this example, a folder contains two messages for which the message ID values are passed in the ROP.

4.3.1 Client Request Buffer

The client request buffer for the **RopDeleteMessages** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.11) example consists of a 23-byte sequence, formatted as follows.

```
0000: 1E 00 00 00 01 02 00 01-00 00 00 0E 8E F1 48 01
0010: 00 00 00 0E 8E C3 02
```

The first five bytes refer to the **RopId** and **LogonID** fields, as specified in [\[MS-OXCROPS\]](#) section 2.2.4.11.1, and the **InputHandleIndex**, **WantAsynchronous**, and **NotifyNonRead** fields of the **RopDeleteMessages** ROP format, as specified in section [2.2.1.11.1](#).

```
0000: 1E 00 00 00 01
```

RopId: 0x1E (**RopDeleteMessages** ROP)

LogonID: 0x00

InputHandleIndex: 0x00. The location where the handle for the messages' parent folder is stored.

WantAsynchronous: 0x00 (FALSE). The ROP is executed synchronously.

NotifyNonRead: 0x01 (TRUE). The client wants a notification if a message was deleted before it was read.

The remaining bytes in the buffer consist of the list of messages to delete.

```
0005: 02 00 01 00 00 00 0E 8E-F1 48 01 00 00 00 0E 8E
0015: C3 02
```

MessageIdCount: 0x0002. This value indicates how many messages are listed for deletion in the **MessageIds** field.

MessageIds:

0001-00000Ee8EF148. MID ([\[MS-OXCDATA\]](#) section 2.2.1.2) of a message to be deleted.

0001-00000E8EC302. MID of a message to be deleted.

4.3.2 Server Responds to Client Request

The server response buffer for the successful **RopDeleteMessages** operation ([\[MS-OXCROPS\]](#) section 2.2.4.11.2) consists of a 7-byte sequence, formatted as follows.

```
0000: 1E 00 00 00 00 00 00
```

RopId: 0x1E (**RopDeleteMessages** ROP)

InputHandleIndex: 0x00. This index is the same as that in the **InputHandleIndex** field specified in the request buffer, as described in section [4.3.1](#).

ReturnValue: 0x00000000. The items were successfully deleted.

PartialCompletion: 0x00 (FALSE). The operation was completed and all messages that were specified in the ROP request were deleted.

4.4 Moving Messages from One Folder to Another

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopMoveCopyMessages** operation, as specified in section [2.2.1.6](#). In this example, a message, specified by its MID ([\[MS-OXCDATA\]](#) section 2.2.1.2), is moved from one folder to another, specified by folder handles.

4.4.1 Client Request Buffer

The client request buffer for the **RopMoveCopyMessages** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.6) example consists of a 16-byte sequence, formatted as follows.

```
0000: 33 00 00 01 01 00 01 00-00 00 0E 8E EC 5D 00 00
```

The first four bytes refer to the **RopId** and **LogonID** fields, as described in [\[MS-OXCROPS\]](#) section 2.2.4.6.1, and the **SourceHandleIndex** and **DestHandleIndex** fields of the **RopMoveCopyMessages** ROP format, as described in section [2.2.1.6.1](#).

```
0000: 33 00 00 01
```

RopId: 0x33 (**RopMoveCopyMessages** ROP)

LogonID: 0x00

SourceHandleIndex: 0x00. The location where the handle for the messages' parent folder is stored.

DestHandleIndex: 0x01. The location where the handle for the destination folder is stored.

The following 10 bytes consist of the list of messages to move.

```
0004: 01 00 01 00 00 00 0E 8E-EC 5D
```

MessageIdCount: 0x0001. This value indicates how many messages are listed for moving in the **MessageIds** field.

MessageIds: 0001-00000E8EEC5D. MID ([\[MS-OXCDATA\]](#) section 2.2.1.2) of the message to be moved.

The final two bytes in the buffer contain the **WantAsynchronous** and **WantCopy** fields, as described in section [2.2.1.6.1](#).

```
000E: 00 00
```

WantAsynchronous: 0x00 (FALSE). The ROP is executed synchronously.

WantCopy: 0x00 (FALSE). The operation is a move rather than a copy.

4.4.2 Server Responds to Client Request

The server response buffer for the successful **RopMoveCopyMessages** operation ([\[MS-OXCROPS\]](#) section 2.2.4.6) consists of a 7-byte sequence formatted as follows.

```
0000: 33 00 00 00 00 00 00
```

RopId: 0x33 (**RopMoveCopyMessages** ROP)

SourceHandleIndex: 0x00. This index is the same as that in the **SourceHandleIndex** field that is specified in the ROP request buffer in section [4.4.1](#).

ReturnValue: 0x00000000. The items were moved.

PartialCompletion: 0x00 (FALSE). The operation was completed and all messages specified in the ROP request were moved.

4.5 Moving a Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopMoveFolder** operation, as specified in [\[MS-OXCROPS\]](#) section 2.2.4.7. In this example, a folder, specified by its FID ([\[MS-OXCDATA\]](#) section 2.2.1.1), is moved to a new location in the folder hierarchy.

4.5.1 Client Request Buffer

The client request buffer for the **RopMoveFolder** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.7) example consists of a 30-byte sequence formatted as follows.

```
0000: 35 00 01 02 01 01 01 00-00 00 0E 8E DF 36 46 00
0010: 6F 00 6C 00 64 00 65 00-72 00 31 00 00 00
```

The first six bytes of the request buffer map to the **RopId** and **LogonID** fields, as described in [\[MS-OXCROPS\]](#) section 2.2.4.7.1, and the **SourceHandleIndex**, **DestHandleIndex**, **WantAsynchronous**, and **UseUnicode** fields of the **RopMoveFolder** ROP format, as described in section [2.2.1.7.1](#).

```
0000: 35 00 01 02 01 01
```

RopId: 0x35 (**RopMoveFolder** ROP)

LogonID: 0x00

SourceHandleIndex: 0x01. The location where the handle for the parent folder of the folder to move is stored.

DestHandleIndex: 0x02. The location where the handle for the destination folder is located.

WantAsynchronous: 0x01 (TRUE). The ROP is executed asynchronously.

UseUnicode: 0x01 (TRUE). The value of the **NewFolderName** field is in Unicode format.

The next eight bytes are the **FolderId** field, as described in section [2.2.1.7.1](#).

```
0006: 01 00 00 00 0E 8E DF 36
```

FolderId: 0001-00000E8EDF36

The remaining 16 bytes of the request buffer specify the new name of the folder.

```
000E: 46 00 6F 00 6C 00 64 00-65 00 72 00 31 00 00 00
```

NewFolderName: "Folder1"

4.5.2 Server Responds to Client Request

The server response buffer for the successful **RopMoveFolder** operation ([\[MS-OXCROPS\]](#) section 2.2.4.7) consists of a 7-byte sequence, formatted as follows.

```
0000: 35 01 00 00 00 00 00
```

RopId: 0x35 (**RopMoveFolder** ROP)

SourceHandleIndex: 0x01. This index is the same as that in the **SourceHandleIndex** field specified in the request buffer in section [4.5.1](#).

ReturnValue: 0x00000000. The folder was successfully moved.

PartialCompletion: 0x00 (FALSE). The operation was fully completed.

4.6 Copying a Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopCopyFolder** operation, as described in [\[MS-OXCROPS\]](#) section 2.2.4.8. In this example, a folder, specified by its FID ([\[MS-OXCADATA\]](#) section 2.2.1.1), is then copied to a new location in the folder hierarchy.

4.6.1 Client Request Buffer

The client request buffer for the **RopCopyFolder** ROP [\[MS-OXCROPS\]](#) section 2.2.4.8) example consists of a 31-byte sequence, formatted as follows.

```
0000: 36 00 00 01 01 01 01 01-00 00 00 0E 8E DF 36 46
0010: 00 6F 00 6C 00 64 00 65-00 72 00 31 00 00 00
```

The first seven bytes of the request buffer map to the **RopId** and **LogonID** fields, as described in [\[MS-OXCROPS\]](#) section 2.2.4.8.1, and the **SourceHandleIndex**, **DestHandleIndex**, **WantAsynchronous**, **WantRecursive**, and **UseUnicode** fields of the **RopCopyFolder** ROP format, as described in section [2.2.1.8.1](#).

```
0000: 36 00 00 01 01 01 01
```

RopId: 0x36 (**RopCopyFolder** ROP)

LogonID: 0x00

SourceHandleIndex: 0x00. The location of where the handle for the parent folder of the folder to copy is stored.

DestHandleIndex: 0x01. The location where the handle for the destination folder is stored.

WantAsynchronous: 0x01 (TRUE). The ROP is executed asynchronously.

WantRecursive: 0x01 (TRUE). The operation recursively copies all subfolders, messages, and properties.

UseUnicode: 0x01 (TRUE). The value of the **NewFolderName** field is in Unicode format.

The next eight bytes are the **FolderId** field, as described in section [2.2.1.8.1](#).

```
0006: 01 00 00 00 0E 8E DF 36
```

FolderId: 0001-00000E8EDF36

The remaining 16 bytes of the request buffer specify the new name of the folder.

```
000E: 46 00 6F 00 6C 00 64 00-65 00 72 00 31 00 00 00
```

NewFolderName: "Folder1"

4.6.2 Server Responds to Client Request

The server response buffer for the successful **RopCopyFolder** operation ([\[MS-OXCROPS\]](#) section 2.2.4.8) consists of a 7-byte sequence, formatted as follows.

```
0000: 36 00 00 00 00 00 00
```

RopId: 0x36 (**RopCopyFolder** ROP)

SourceHandleIndex: 0x00. This index is the same as that in the **SourceHandleIndex** field specified in the request buffer in section [4.6.1](#).

ReturnValue: 0x00000000. The folder was moved.

PartialCompletion: 0x00 (FALSE). The operation was completed.

4.7 Getting the List of Subfolders Within a Message Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopGetHierarchyTable** operation, as described in [\[MS-OXCROPS\]](#) section 2.2.4.13. For information about tables, see [\[MS-OXCTABL\]](#).

4.7.1 Client Request Buffer

The client request buffer for the **RopGetHierarchyTable** ([\[MS-OXCROPS\]](#) section 2.2.4.13) example consists of a 5-byte sequence, formatted as follows.

```
0000: 04 00 01 02 00
```

RopId: 0x04 (**RopGetHierarchyTable** ROP)

LogonID: 0x00

InputHandleIndex: 0x01. The location where the handle for the folder to retrieve the hierarchy table is stored.

OutputHandleIndex: 0x02. The location where the handle for the hierarchy table will be stored.

TableFlags: 0x00.

4.7.2 Server Responds to Client Request

The server response buffer for the successful **RopGetHierarchyTable** operation ([\[MS-OXCROPS\]](#) section 2.2.4.13) consists of a 10-byte sequence, formatted as follows.

```
0000: 04 02 00 00 00 00 15 00-00 00
```

RopId: 0x04 (**RopGetHierarchyTable** ROP)

OutputHandleIndex: 0x02. This index is the same as that in the **OutputHandleIndex** field specified in the request buffer in section [4.7.1](#).

Return Value: 0x00000000. The hierarchy table was retrieved.

RowCount: 0x00000015. The table contains 21 rows.

4.8 Setting the Search Criteria for a Search Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopSetSearchCriteria** operation, as described in [\[MS-OXCROPS\]](#) section 2.2.4.4. The search folder (2) is referred to by the **InputHandleIndex** field, and the search criteria filter specifies restrictions (2) that limit the items in the search folder (2)—in this case, mail items for which the **PidTagImportance** property ([\[MS-OXCMSG\]](#) section 2.2.1.11) is set to 0x00000002 (High). For more details about the structure of a restriction (2), see [\[MS-OXCDATA\]](#) section 2.12.

4.8.1 Client Request Buffer

The client request buffer for the **RopSetSearchCriteria** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.4) example consists of a 316-byte sequence, formatted as follows.

```
0000: 30 00 01 29 01 00 02 00-00 07 00 02 03 02 00 01
0010: 00 1F 00 1A 00 1F 00 1A-00 49 00 50 00 4D 00 2E
0020: 00 41 00 70 00 70 00 6F-00 69 00 6E 00 74 00 6D
0030: 00 65 00 6E 00 74 00 00-00 02 03 02 00 01 00 1F
0040: 00 1A 00 1F 00 1A 00 49-00 50 00 4D 00 2E 00 43
0050: 00 6F 00 6E 00 74 00 61-00 63 00 74 00 00 00 02
0060: 03 02 00 01 00 1F 00 1A-00 1F 00 1A 00 49 00 50
0070: 00 4D 00 2E 00 44 00 69-00 73 00 74 00 4C 00 69
0080: 00 73 00 74 00 00 00 02-03 02 00 01 00 1F 00 1A
0090: 00 1F 00 1A 00 49 00 50-00 4D 00 2E 00 41 00 63
00A0: 00 74 00 69 00 76 00 69-00 74 00 79 00 00 00 02
00B0: 03 02 00 01 00 1F 00 1A-00 1F 00 1A 00 49 00 50
00C0: 00 4D 00 2E 00 53 00 74-00 69 00 63 00 6B 00 79
00D0: 00 4E 00 6F 00 74 00 65-00 00 00 02 03 00 00 01
00E0: 00 1F 00 1A 00 1F 00 1A-00 49 00 50 00 4D 00 2E
00F0: 00 54 00 61 00 73 00 6B-00 00 00 02 03 02 00 01
0100: 00 1F 00 1A 00 1F 00 1A-00 49 00 50 00 4D 00 2E
0110: 00 54 00 61 00 73 00 6B-00 2E 00 00 00 00 01 00
0120: 04 04 03 00 17 00 03 00-17 00 02 00 00 00 01 00
0130: 01 00 00 00 00 00 00 14 88-2A 00 02 00
```

The first three bytes of the request buffer map to the **RopId** and **LogonID** fields, as described in [\[MS-OXCROPS\]](#) section 2.2.4.4.1, and the **InputHandleIndex** field, as specified in section [2.2.1.4.1](#), of the **RopSetSearchCriteria** ROP format.

```
0000: 30 00 01
```

RopId: 0x30 (**RopSetSearchCriteria** ROP)

LogonID: 0x00

InputHandleIndex: 0x01. The location where the handle for the search folder (2) to configure is stored.

The next 299 bytes comprise the restriction (2) that defines the search criteria for the search folder (2), broken down in further detail as follows.

```
0003: 29 01 00 02 00 00 07 00-02 03 02 00 01 00 1F 00
0013: 1A 00 1F 00 1A 00 49 00-50 00 4D 00 2E 00 41 00
0023: 70 00 70 00 6F 00 69 00-6E 00 74 00 6D 00 65 00
0033: 6E 00 74 00 00 00 02 03-02 00 01 00 1F 00 1A 00
0043: 1F 00 1A 00 49 00 50 00-4D 00 2E 00 43 00 6F 00
0053: 6E 00 74 00 61 00 63 00-74 00 00 00 02 03 02 00
0063: 01 00 1F 00 1A 00 1F 00-1A 00 49 00 50 00 4D 00
0073: 2E 00 44 00 69 00 73 00-74 00 4C 00 69 00 73 00
0083: 74 00 00 00 02 03 02 00-01 00 1F 00 1A 00 1F 00
0093: 1A 00 49 00 50 00 4D 00-2E 00 41 00 63 00 74 00
00A3: 69 00 76 00 69 00 74 00-79 00 00 00 02 03 02 00
00B3: 01 00 1F 00 1A 00 1F 00-1A 00 49 00 50 00 4D 00
00C3: 2E 00 53 00 74 00 69 00-63 00 6B 00 79 00 4E 00
00D3: 6F 00 74 00 65 00 00 00-02 03 00 00 01 00 1F 00
00E3: 1A 00 1F 00 1A 00 49 00-50 00 4D 00 2E 00 54 00
00F3: 61 00 73 00 6B 00 00 00-02 03 02 00 01 00 1F 00
0103: 1A 00 1F 00 1A 00 49 00-50 00 4D 00 2E 00 54 00
0113: 61 00 73 00 6B 00 2E 00-00 00 00 01 00 04 04 03
0123: 00 17 00 03 00 17 00 02-00 00 00
```

RestrictionDataSize: 0x0129. The size of the restriction (2) block is 297 bytes.

RestrictionData: Bytes 0005-012A, which translate into the following restriction (2):

RestrictType: 0x00 (**RES_AND**)

RestrictCount: 0x0002

RestrictType: 0x00 (**RES_AND**)

RestrictCount: 0x0007

RestrictType: 0x02 (**RES_NOT**)

RestrictType: 0x03 (**RES_CONTENT**)

FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)

PropTag1: 0x001A001F (**PidTagMessageClass** property ([\[MS-OXCMSG\]](#) section 2.2.1.3))

PropTag2: 0x001A001F (**PidTagMessageClass** property)

PropRule: "IPM.Appointment"

RestrictType: 0x02 (**RES_NOT**)

RestrictType: 0x03 (**RES_CONTENT**)

FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)

PropTag1: 0x001A001F" (**PidTagMessageClass** property)

PropTag2: "0x001A001F " (**PidTagMessageClass** property)
PropRule: "IPM.Contact"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.DistList"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.Activity"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.StickyNote"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010000 (FL_FULLSTRING | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.Task"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)

PropTag2: 0x001A001F (**PidTagMessageClass** property)

PropRule: "IPM.Task. "

RestrictType: 0x00 (**RES_AND**)

RestrictCount: 0x0001

RestrictType: 0x04 (**RES_PROPERTY**)

RelOp: 0x04 (RELOP_EQ)

PropTag1: 0x00170003 (**PidTagImportance** property ([\[MS-OXCMSG\]](#) section 2.2.1.11))

PropTag2: 0x00170003 (**PidTagImportance** property)

PropValue: 0x00000002

A shorthand description of the restriction (2) is as follows:

(**PidTagMessageClass** is not equal to "IPM.Appointment" **AND**

PidTagMessageClass is not equal to "IPM.Contact" **AND**

PidTagMessageClass is not equal to "IPM.DistList" **AND**

PidTagMessageClass is not equal to "IPM.Activity" **AND**

PidTagMessageClass is not equal to "IPM.StickyNote" **AND**

PidTagMessageClass is not equal to "IPM.Task" **AND**

PidTagMessageClass is not equal to "IPM.Task.")

The next 10 bytes consist of the **FolderIdCount** and **FolderIds** fields, as described in section [2.2.1.4.1](#).

```
012E: 01 00 01 00 00 00 00 00-14 88
```

FolderIdCount: 0x0001. The number of folders within the scope of the search folder (2).

FolderIds: 0001-000000001488. Identifies the folder to be searched.

The remaining four bytes represent the **SearchFlags** field, as described in section [2.2.1.4.1](#).

```
0138: 2A 00 02 00
```

SearchFlags: 0x0002002A (RESTART_SEARCH | SHALLOW_SEARCH | BACKGROUND_SEARCH | NON_CONTENT_INDEXED_SEARCH)

4.8.2 Server Responds to Client Request

The server response buffer for the **RopSetSearchCriteria** ROP ([\[MS-OXCROPS\]](#) section 2.2.4.4) operation consists of a 6-byte sequence, formatted as follows.

```
0000: 30 01 00 00 00 00
```

RopId: 0x30 (**RopSetSearchCriteria** ROP)

InputHandleIndex: 0x01. This index is the same as that in the **InputHandleIndex** specified in the request buffer in section [4.8.1](#).

ReturnValue: 0x00000000. The search criteria were set on the folder.

4.9 Getting the Search Criteria for a Search Folder

The following example describes the content of the ROP request buffer and ROP response buffer for a successful **RopGetSearchCriteria** operation, as specified in [\[MS-OXCROPS\]](#) section 2.2.4.5. The search folder (2) is referred to by the **InputHandleIndex** field, and the search criteria filter that is returned specifies restrictions (2) that limit the items in the search folder (2)—in this case, mail items for which the **PidTagImportance** property ([\[MS-OXCMSG\]](#) section 2.2.1.11) is set to 0x00000002 (High). For information about the structure of a restriction (2), see [\[MS-OXCDATA\]](#) section 2.12.

4.9.1 Client Request Buffer

The client request buffer for the **RopGetSearchCriteria** ([\[MS-OXCROPS\]](#) section 2.2.4.5) example consists of a sequence of 6-byte sequence, formatted as follows.

```
0000: 31 00 00 01 01 00
```

RopId: 0x31 (**RopGetSearchCriteria** ROP)

LogonID: 0x00

InputHandleIndex: 0x00. The location where the handle for the search folder (2) to query for search criteria is stored.

UseUnicode: 0x01 (TRUE). The restriction (2) data in the response is expected to be in Unicode strings.

IncludeRestriction: 0x01 (TRUE). The server response is expected to include the restriction (2) data for the search folder (2).

IncludeFolders: 0x00 (FALSE). The server response is not expected to include the set of folders within the search scope.

4.9.2 Server Responds to Client Request

The server response buffer for the successful **RopGetSearchCriteria** operation ([\[MS-OXCROPS\]](#) section 2.2.4.5) consists of a 312-byte sequence, formatted as follows.

```
0000: 31 00 00 00 00 00 29 01-00 02 00 00 07 00 02 03
0010: 02 00 01 00 1F 00 1A 00-1F 00 1A 00 49 00 50 00
0020: 4D 00 2E 00 41 00 70 00-70 00 6F 00 69 00 6E 00
0030: 74 00 6D 00 65 00 6E 00-74 00 00 00 02 03 02 00
0040: 01 00 1F 00 1A 00 1F 00-1A 00 49 00 50 00 4D 00
0050: 2E 00 43 00 6F 00 6E 00-74 00 61 00 63 00 74 00
0060: 00 00 02 03 02 00 01 00-1F 00 1A 00 1F 00 1A 00
```

```

0070: 49 00 50 00 4D 00 2E 00-44 00 69 00 73 00 74 00
0080: 4C 00 69 00 73 00 74 00-00 00 02 03 02 00 01 00
0090: 1F 00 1A 00 1F 00 1A 00-49 00 50 00 4D 00 2E 00
00A0: 41 00 63 00 74 00 69 00-76 00 69 00 74 00 79 00
00B0: 00 00 02 03 02 00 01 00-1F 00 1A 00 1F 00 1A 00
00C0: 49 00 50 00 4D 00 2E 00-53 00 74 00 69 00 63 00
00D0: 6B 00 79 00 4E 00 6F 00-74 00 65 00 00 00 02 03
00E0: 00 00 01 00 1F 00 1A 00-1F 00 1A 00 49 00 50 00
00F0: 4D 00 2E 00 54 00 61 00-73 00 6B 00 00 00 02 03
0100: 02 00 01 00 1F 00 1A 00-1F 00 1A 00 49 00 50 00
0110: 4D 00 2E 00 54 00 61 00-73 00 6B 00 2E 00 00 00
0120: 00 01 00 04 04 03 00 17-00 03 00 17 00 02 00 00
0130: 00 00 00 00 01 00 00 00

```

The first six bytes contain the **RopId** and **InputHandleIndex** fields, as described in [\[MS-OXCROPS\]](#) section 2.2.4.5.2, and the **ReturnValue** field as described in section [2.2.1.5.2](#):

```
0000: 31 00 00 00 00 00
```

RopId: 0x31 (**RopGetSearchCriteria** ROP)

InputHandleIndex: 0x00. This index is the same as that in the **InputHandleIndex** field specified in the request buffer in section [4.9.1](#).

ReturnValue: 0x00000000. The search criteria for the search folder (2) were retrieved.

The next 299 bytes comprise the restrictions (2) that define the search criteria for the search folder (2), broken down in further detail as follows.

```

0006: 29 01 00 02 00 00 07 00-02 03 02 00 01 00 1F 00
0016: 1A 00 1F 00 1A 00 49 00-50 00 4D 00 2E 00 41 00
0026: 70 00 70 00 6F 00 69 00-6E 00 74 00 6D 00 65 00
0036: 6E 00 74 00 00 00 02 03-02 00 01 00 1F 00 1A 00
0046: 1F 00 1A 00 49 00 50 00-4D 00 2E 00 43 00 6F 00
0056: 6E 00 74 00 61 00 63 00-74 00 00 00 02 03 02 00
0066: 01 00 1F 00 1A 00 1F 00-1A 00 49 00 50 00 4D 00
0076: 2E 00 44 00 69 00 73 00-74 00 4C 00 69 00 73 00
0086: 74 00 00 00 02 03 02 00-01 00 1F 00 1A 00 1F 00
0096: 1A 00 49 00 50 00 4D 00-2E 00 41 00 63 00 74 00
00A6: 69 00 76 00 69 00 74 00-79 00 00 00 02 03 02 00
00B6: 01 00 1F 00 1A 00 1F 00-1A 00 49 00 50 00 4D 00
00C6: 2E 00 53 00 74 00 69 00-63 00 6B 00 79 00 4E 00
00D6: 6F 00 74 00 65 00 00 00-02 03 00 00 01 00 1F 00
00E6: 1A 00 1F 00 1A 00 49 00-50 00 4D 00 2E 00 54 00
00F6: 61 00 73 00 6B 00 00 00-02 03 02 00 01 00 1F 00
0106: 1A 00 1F 00 1A 00 49 00-50 00 4D 00 2E 00 54 00
0116: 61 00 73 00 6B 00 2E 00-00 00 00 01 00 04 04 03
0126: 00 17 00 03 00 17 00 02-00 00 00

```

RestrictionDataSize: 0x0129. The size of the restriction (2) block, which is 297 bytes.

RestrictionData: Bytes 0008-0130, which translate into the following restriction (2):

RestrictType: 0x00 (**RES_AND**)
RestrictCount: 0x0002
RestrictType: 0x00 (**RES_AND**)
RestrictCount: 0x0007
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property ([\[MS-OXCMSG\]](#) section 2.2.1.3))
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.Appointment"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.Contact"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.DistList"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.Activity"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)

FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.StickyNote"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010000 (FL_FULLSTRING | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.Task"
RestrictType: 0x02 (**RES_NOT**)
RestrictType: 0x03 (**RES_CONTENT**)
FuzzyLevel: 0x00010002 (FL_PREFIX | FL_IGNORECASE)
PropTag1: 0x001A001F (**PidTagMessageClass** property)
PropTag2: 0x001A001F (**PidTagMessageClass** property)
PropRule: "IPM.Task."
RestrictType: 0x00 (**RES_AND**)
RestrictCount: 0x0001
RestrictType: 0x04(**RES_PROPERTY**)
RelOp: 0x04 (RELOP_EQ)
PropTag1: 0x00170003 (**PidTagImportance** property ([\[MS-OXCMSG\]](#) section 2.2.1.11))
PropTag2: 0x00170003 (**PidTagImportance** property)
PropValue: 0x00000002

A shorthand pseudocode description of the restriction (2) is as follows:

(**PidTagMessageClass** is not equal to "IPM.Appointment" **AND**
PidTagMessageClass is not equal to "IPM.Contact" **AND**
PidTagMessageClass is not equal to "IPM.DistList" **AND**
PidTagMessageClass is not equal to "IPM.Activity" **AND**
PidTagMessageClass is not equal to "IPM.StickyNote" **AND**
PidTagMessageClass is not equal to "IPM.Task" **AND**

PidTagMessageClass is not equal to "IPM.Task. ")

The final seven bytes of the server response buffer contain the **LogonID** field, as described in [\[MS-OXCROPS\]](#) section 2.2.4.5.3, and the **FolderIdCount** and **SearchFlags** fields, as described in section [2.2.1.5.2](#).

```
0131: 00 00 00 01 00 00 00
```

LogonID: 0x00. This is the same value as the **LogonID** field specified in the request buffer in section [4.9.1](#).

FolderIdCount: 0x0000.

SearchFlags: 0x00000001 (SEARCH_RUNNING).

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to this specification. General security considerations that pertain to the underlying ROP-based transport apply.

5.2 Index of Security Parameters

None.

6 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® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010

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 2.2.1.1.2:](#) Exchange 2003 and Exchange 2007 return zero (FALSE) in the **HasRules** field, even when there are rules (4) on the **Inbox folder**.

[<2> Section 2.2.1.2.2:](#) Exchange 2010 returns zero (FALSE) in the **IsExistingFolder** field when the named folder already exists.

[<3> Section 2.2.1.2.2:](#) Exchange 2003 and Exchange 2007 return zero (FALSE) in the **HasRules** field, even when there are rules (4) on the Inbox folder.

[<4> Section 2.2.1.14.1:](#) Exchange 2003 and Exchange 2007 do not support the ConversationMembers bit.

[<5> Section 2.2.2.2.6:](#) Outlook 2010 does not use the **PidTagAccessControllistData** property ([MS-OXCPERM] section 2.2.1.3).

[<6> Section 3.2.5.1:](#) If the specified folder has been hard deleted, Exchange 2007 does not fail the **PopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1), but no folder is opened.

[<7> Section 3.2.5.2:](#) Exchange 2007 returns ecAccessdenied. Exchange 2010 returns ecNoCreateSubfolderRight.

[<8> Section 3.2.5.2:](#) Exchange 2007 returns ecNotSupported. Exchange 2010 returns ecError.

[<9> Section 3.2.5.4:](#) Exchange 2007 silently ignores invalid bits and does not return the ecInvalidParam error code.

[<10> Section 3.2.5.13:](#) Exchange 2007 ignores invalid bits instead of failing the ROP.

[<11> Section 3.2.5.14:](#) Exchange 2003 fails the ROP with an error code of ecInvalidParam. Exchange 2007 ignores the ConversationMembers bit.

[<12> Section 3.2.5.14:](#) Exchange 2007 ignores invalid bits instead of failing the ROP.

7 Change Tracking

This section identifies changes that were made to the [MS-OXCFOLD] protocol document between the August 2011 and October 2011 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
2.2.1.14.1 RopGetContentsTable ROP Request Buffer	Added a product behavior note to specify that Exchange 2003 and Exchange 2007 do not support the ConversationMembers bit.	Y	New product behavior note added.
3.1.4.10 Getting a Contents Table	Added details about how client gets a contents table that shows conversation view.	N	Content updated.
3.2.5.14 Processing a RopGetContentsTable ROP Request	Added a product behavior note to specify Exchange 2003 and Exchange 2007 behavior when the ConversationMembers bit is set.	Y	New product behavior note added.
3.2.5.14 Processing a RopGetContentsTable ROP Request	Added details about the server response when the ConversationMembers bit of the TableFlags field is set.	N	Content updated.

8 Index

A

Abstract data model
[client](#) 31
[server](#) 35
Abstract data model - client
[contents table](#) 31
[hierachy table](#) 31
[Applicability](#) 10

C

[Capability negotiation](#) 10
[Change tracking](#) 67
Client
[abstract data model](#) 31
[initialization](#) 32
[other local events](#) 35
[timer events](#) 35
[timers](#) 32
Client - abstract data model
[contents table](#) 31
[hierachy table](#) 31
Client - higher-layer triggered events
[copying a folder and its contents](#) 33
[creating a folder](#) 32
[deleting a folder](#) 32
[deleting the contents of a folder](#) 34
[getting a contents table](#) 34
[getting a hierachy table](#) 34
[getting details about a search folder](#) 33
[moving a folder and its contents](#) 33
[opening a folder](#) 32
[setting up a folder](#) 32
Client - message processing
[processing ROPs asynchronously](#) 35
[releasing a Server object](#) 35
Client - sequencing rules
[processing ROPs aynchronously](#) 35
[releasing a Server object](#) 35
Copying a folder example
[client request buffer](#) 53
[overview](#) 53
[server responds to client request](#) 54
Creating a new folder example
[client request buffer](#) 46
[overview](#) 46
[server responds to client request](#) 47

D

Data model - abstract
[client](#) 31
[server](#) 35
Deleting an existing folder example
[client request buffer](#) 48
[overview](#) 48
[server responds to client request](#) 49

Deleting messages within a folder example
[client request buffer](#) 49
[overview](#) 49
[server responds to client request](#) 50

E

Examples
[copying a folder](#) 53
[creating a new folder](#) 46
[deleting an existing folder](#) 48
[deleting messages within a folder](#) 49
[getting the list of subfolders within a message folder](#) 54
[getting the search criteria for a search folder](#) 59
[moving a folder](#) 51
[moving messages from one folder to another](#) 50
[overview](#) 46
[setting the search criteria for a search folder](#) 55

F

[Fields - vendor-extensible](#) 10
Folder object properties
[Folder object specific properties](#) 28
[general properties](#) 28
[Folder Object Properties message](#) 27
[Folder object specific properties](#) 28

G

[General Folder object properties](#) 28
Getting the list of subfolders within a message folder example
[client request buffer](#) 54
[overview](#) 54
[server responds to client request](#) 54
Getting the search criteria for a search folder example
[client request buffer](#) 59
[overview](#) 59
[server responds to client request](#) 59
[Glossary](#) 7

H

Higher-layer triggered events
[server](#) 36
Higher-layer triggered events - client
[copying a folder and its contents](#) 33
[creating a folder](#) 32
[deleting a folder](#) 32
[deleting the contents of a folder](#) 34
[getting a contents table](#) 34
[getting a hierachy table](#) 34
[getting details about a search folder](#) 33
[moving a folder and its contents](#) 33
[opening a folder](#) 32

[setting up a folder](#) 32

I

[Implementer - security considerations](#) 64

[Index of security parameters](#) 64

[Informative references](#) 9

Initialization

[client](#) 32

[server](#) 36

[Introduction](#) 7

M

Message processing - client

[processing ROPs asynchronously](#) 35

[releasing a Server object](#) 35

Message processing - server

[processing a RopCopyFolder ROP request](#) 41

[processing a RopCreateFolder ROP request](#) 37

[processing a RopDeleteFolder ROP request](#) 37

[processing a RopDeleteMessages ROP request](#) 43

[processing a RopEmptyFolder ROP request](#) 42

[processing a RopGetContentsTable ROP request](#)
45

[processing a RopGetHierarchyTable ROP request](#)
44

[processing a RopGetSearchCriteria ROP request](#)
40

[processing a RopHardDeleteMessages ROP
request](#) 44

[processing a
RopHardDeleteMessagesAndSubfolders ROP
request](#) 43

[processing a RopMoveCopyMessages ROP request](#)
40

[processing a RopMoveFolder ROP request](#) 41

[processing a RopOpenFolder ROP request](#) 36

[processing a RopSetSearchCriteria ROP request](#)
38

Messages

[Folder Object Properties](#) 27

[ROPs](#) 11

[transport](#) 11

Moving a folder example

[client request buffer](#) 52

[overview](#) 51

[server responds to client request](#) 52

Moving messages from one folder to another
example

[client request buffer](#) 50

[overview](#) 50

[server responds to client request](#) 51

N

[Normative references](#) 8

O

Other local events

[client](#) 35

[server](#) 45

[Overview \(synopsis\)](#) 9

P

[Parameters - security index](#) 64

[Preconditions](#) 10

[Prerequisites](#) 10

[Product behavior](#) 65

R

References

[informative](#) 9

[normative](#) 8

[Relationship to other protocols](#) 9

[RopCopyFolder ROP](#) 21

[RopCreateFolder ROP](#) 12

[RopDeleteFolder ROP](#) 14

[RopDeleteMessages ROP](#) 23

[RopEmptyFolder ROP](#) 22

[RopGetContentsTable ROP](#) 26

[RopGetHierarchyTable ROP](#) 25

[RopGetSearchCriteria ROP](#) 17

[RopHardDeleteMessages ROP](#) 24

[RopHardDeleteMessagesAndSubfolders ROP](#) 22

[RopMoveCopyMessages ROP](#) 19

[RopMoveFolder ROP](#) 20

[RopOpenFolder ROP](#) 11

ROPs

[RopCopyFolder ROP](#) 21

[RopCreateFolder ROP](#) 12

[RopDeleteFolder ROP](#) 14

[RopDeleteMessages ROP](#) 23

[RopEmptyFolder ROP](#) 22

[RopGetContentsTable ROP](#) 26

[RopGetHierarchyTable ROP](#) 25

[RopGetSearchCriteria ROP](#) 17

[RopHardDeleteMessages ROP](#) 24

[RopHardDeleteMessagesAndSubfolders ROP](#) 22

[RopMoveCopyMessages ROP](#) 19

[RopMoveFolder ROP](#) 20

[RopOpenFolder ROP](#) 11

[RopSetSearchCriteria ROP](#) 15

[ROPs message](#) 11

[RopSetSearchCriteria ROP](#) 15

S

Security

[implementer considerations](#) 64

[parameter index](#) 64

Sequencing rules - client

[processing ROPs asynchronously](#) 35

[releasing a Server object](#) 35

Sequencing rules - server

[processing a RopCopyFolder ROP request](#) 41

[processing a RopCreateFolder ROP request](#) 37

[processing a RopDeleteFolder ROP request](#) 37

[processing a RopDeleteMessages ROP request](#) 43

[processing a RopEmptyFolder ROP request](#) 42

- [processing a RopGetContentsTable ROP request](#) 45
- [processing a RopGetHierarchyTable ROP request](#) 44
- [processing a RopGetSearchCriteria ROP request](#) 40
- [processing a RopHardDeleteMessages ROP request](#) 44
- [processing a RopHardDeleteMessagesAndSubfolders ROP request](#) 43
- [processing a RopMoveCopyMessages ROP request](#) 40
- [processing a RopMoveFolder ROP request](#) 41
- [processing a RopOpenFolder ROP request](#) 36
- [processing a RopSetSearchCriteria ROP request](#) 38
- Server
 - [abstract data model](#) 35
 - [higher-layer triggered events](#) 36
 - [initialization](#) 36
 - [other local events](#) 45
 - [timer events](#) 45
 - [timers](#) 36
- Server - message processing
 - [processing a RopCopyFolder ROP request](#) 41
 - [processing a RopCreateFolder ROP request](#) 37
 - [processing a RopDeleteFolder ROP request](#) 37
 - [processing a RopDeleteMessages ROP request](#) 43
 - [processing a RopEmptyFolder ROP request](#) 42
 - [processing a RopGetContentsTable ROP request](#) 45
 - [processing a RopGetHierarchyTable ROP request](#) 44
 - [processing a RopGetSearchCriteria ROP request](#) 40
 - [processing a RopHardDeleteMessages ROP request](#) 44
 - [processing a RopHardDeleteMessagesAndSubfolders ROP request](#) 43
 - [processing a RopMoveCopyMessages ROP request](#) 40
 - [processing a RopMoveFolder ROP request](#) 41
 - [processing a RopOpenFolder ROP request](#) 36
 - [processing a RopSetSearchCriteria ROP request](#) 38
- Server - sequencing rules
 - [processing a RopCopyFolder ROP request](#) 41
 - [processing a RopCreateFolder ROP request](#) 37
 - [processing a RopDeleteFolder ROP request](#) 37
 - [processing a RopDeleteMessages ROP request](#) 43
 - [processing a RopEmptyFolder ROP request](#) 42
 - [processing a RopGetContentsTable ROP request](#) 45
 - [processing a RopGetHierarchyTable ROP request](#) 44
 - [processing a RopGetSearchCriteria ROP request](#) 40
 - [processing a RopHardDeleteMessages ROP request](#) 44
- [processing a RopHardDeleteMessagesAndSubfolders ROP request](#) 43
- [processing a RopMoveCopyMessages ROP request](#) 40
- [processing a RopOpenFolder ROP request](#) 36
- [processing a RopSetSearchCriteria ROP request](#) 38
- [processing a RopMoveFolder ROP request](#) 41
- Setting the search criteria for a search folder
 - example
 - [client request buffer](#) 55
 - [overview](#) 55
 - [server responds to client request](#) 58
- [Standards assignments](#) 10

T

- Timer events
 - [client](#) 35
 - [server](#) 45
- Timers
 - [client](#) 32
 - [server](#) 36
- [Tracking changes](#) 67
- [Transport](#) 11
- Triggered events - client
 - [copying a folder and its contents](#) 33
 - [creating a folder](#) 32
 - [deleting a folder](#) 32
 - [deleting the contents of a folder](#) 34
 - [getting a contents table](#) 34
 - [getting a hierarchy table](#) 34
 - [getting details about a search folder](#) 33
 - [moving a folder and its contents](#) 33
 - [opening a folder](#) 32
 - [setting up a folder](#) 32
- Triggered events - higher-layer
 - [server](#) 36

V

- [Vendor-extensible fields](#) 10
- [Versioning](#) 10