[MS-OXCFOLD]:

Folder Object Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional

development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1		Initial Availability.
4/25/2008	0.2		Revised and updated property names and other technical content.
6/27/2008	1.0		Initial Release.
8/6/2008	1.01		Revised and edited technical content.
9/3/2008	1.02		Revised and edited technical content.
12/3/2008	1.03		Revised and edited technical content.
3/4/2009	1.04		Revised and edited technical content.
4/10/2009	2.0		Updated technical content and applicable product releases.
7/15/2009	3.0	Major	Revised and edited for technical content.
11/4/2009	4.0.0	Major	Updated and revised the technical content.
2/10/2010	5.0.0	Major	Updated and revised the technical content.
5/5/2010	5.0.1	Editorial	Revised and edited the technical content.
8/4/2010	6.0	Major	Significantly changed the technical content.
11/3/2010	7.0	Major	Significantly changed the technical content.
3/18/2011	7.0	No change	No changes to the meaning, language, and formatting of the technical content.
8/5/2011	8.0	Major	Significantly changed the technical content.
10/7/2011	9.0	Major	Significantly changed the technical content.
1/20/2012	10.0	Major	Significantly changed the technical content.
4/27/2012	11.0	Major	Significantly changed the technical content.
7/16/2012	11.0	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	12.0	Major	Significantly changed the technical content.
2/11/2013	13.0	Major	Significantly changed the technical content.
7/26/2013	14.0	Major	Significantly changed the technical content.
11/18/2013	14.1	Minor	Clarified the meaning of the technical content.
2/10/2014	14.1	No Change	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	14.1	No Change	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	14.2	Minor	Clarified the meaning of the technical content.

Date	Revision History	Revision Class	Comments
10/30/2014	15.0	Major	Significantly changed the technical content.
3/16/2015	16.0	Major	Significantly changed the technical content.
5/26/2015	17.0	Major	Significantly changed the technical content.

Table of Contents

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 can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

access control list (ACL): A list of access control entries (ACEs) that collectively describe the security rules for authorizing access to some resource; for example, an object or set of objects.

active replica: A name given to a server that hosts content and is expected to serve that content to clients.

ASCII: The American Standard Code for Information Interchange (ASCII) is an 8-bit character-encoding scheme based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text. ASCII refers to a single 8-bit ASCII character or an array of 8-bit ASCII characters with the high bit of each character set to zero.

code page: An ordered set of characters of a specific script in which a numerical index (code-point value) is associated with each character. Code pages are a means of providing support for character sets (1) and keyboard layouts used in different countries. Devices such as the display and keyboard can be configured to use a specific code page and to switch from one code page (such as the United States) to another (such as Portugal) at the user's request.

contents table: A **Table object** whose rows represent the **Message objects** that are contained in a **Folder object**.

conversation: A single representation of a send/response series of email messages. A conversation appears in the Inbox as one unit and allows the user to view and read the series of related email messages in a single effort.

Coordinated Universal Time (UTC): A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC–0 (or GMT).

Drafts folder: A **special folder** that is the default location for **Message objects** that have been saved but not sent.

entry ID: See EntryID.

FAI contents table: A table of **folder associated information (FAI)** Message objects that are stored in a Folder object.

folder associated information (FAI): A collection of **Message objects** that are stored in a Folder object and are typically hidden from view by email applications. An FAI Message object is used to store a variety of settings and auxiliary data, including forms, views, calendar options, favorites, and category lists.

Folder object: A messaging construct that is typically used to organize data into a hierarchy of objects containing Message objects and **folder associated information (FAI)** Message objects.

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.

ghosted folder: A folder whose contents are located on another server.

handle: Any token that can be used to identify and access an object such as a device, file, or a window.

hard delete: A process that removes an item permanently from the system. If an item is hard deleted, a server does not retain a back-up copy of the item and a client cannot access or restore the item. See also **soft delete**.

hierarchy table: A **Table object** whose rows represent the **Folder objects** that are contained in another Folder object.

Inbox folder: A **special folder** that is the default location for **Message objects** received by a user or resource.

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

Logon object: A **Server object** that provides access to a private **mailbox** or a **public folder**. A client obtains a Logon object by issuing a RopLogon **remote operation (ROP)** to a server.

mailbox: A **message store** that contains email, calendar items, and other **Message objects** for a single recipient.

Message object: A set of properties that represents an email message, appointment, contact, or other type of personal-information-management object. In addition to its own properties, a Message object contains recipient properties that represent the addressees to which it is addressed, and an attachments table that represents any files and other Message objects that are attached to it.

message store: A unit of containment for a single hierarchy of Folder objects, such as a mailbox or public folders.

messaging object: An object that exists in a **mailbox**. It can be only a **Folder object** or a **Message object**.

non-read receipt: A message that is generated when an email message is deleted at the expiration of a time limit or due to other client-specific criteria.

normal message: A message that is not a **folder associated information (FAI)** message.

permission: A rule that is associated with an object and that regulates which users can gain access to the object and in what manner. See also rights.

property type: A 16-bit quantity that specifies the data type of a property value.

public folder: A **Folder object** that is stored in a location that is publicly available.

read receipt: An email message that is sent to the sender of a message to indicate that a message recipient received the message.

remote operation (ROP): An operation that is invoked against a server. Each ROP represents an action, such as delete, send, or query. A ROP is contained in a **ROP buffer** for transmission over the wire.

replica: A server that hosts an instance of a message item in a folder.

restriction: A filter used to map some domain into a subset of itself, by passing only those items from the domain that match the filter. Restrictions can be used to filter existing **Table objects** or to define new ones, such as **search folder (2)** or rule criteria.

Root folder: The **special folder** that is the top-level folder in a message store hierarchy. It contains all other **Folder objects** in that message store.

ROP buffer: A structure containing an array of bytes that encode a **remote operation (ROP)**. The first byte in the buffer identifies the ROP. This byte is followed by ROP-specific fields. Multiple ROP buffers can be packed into a single remote procedure call (RPC) request or response.

ROP request: See **ROP request buffer**.

ROP request buffer: A **ROP buffer** that a client sends to a server to be processed.

ROP response: See **ROP response buffer**.

ROP response buffer: A **ROP buffer** that a server sends to a client to be processed.

rule: An item that defines a condition and an action. The condition is evaluated for each **Message object** as it is delivered, and the action is executed if the new Message object matches the condition.

search criteria: A criteria used to determine which messages are included in a folder with specific characteristics. It is composed of a restriction, which is the filter to be applied, and a search scope, which are the folders that contain the content to search.

search folder: (1) A collection of related items to be crawled by a search service.

(2) A **Folder object** that provides a means of querying for items that match certain criteria. The search folder includes the search folder definition message and the **search folder container**.

search folder container: A **Folder object** that is created according to the specifications in the definition message. It is in the Finder folder of the message database.

Server object: An object on a server that is used as input or created as output for **remote operations (ROPs)**.

Server object handle: A 32-bit value that identifies a **Server object**.

Server object handle table: An array of 32-bit handles that are used to identify input and output **Server objects** for **ROP requests** and **ROP responses**.

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

soft delete: A process that removes an item from the system, but not permanently. If an item is soft deleted, a server retains a back-up copy of the item and a client can access, restore, or permanently delete the item. See also **hard delete**.

special folder: One of a default set of **Folder objects** that can be used by an implementation to store and retrieve user data objects.

stream: An element of a compound file, as described in [MS-CFB]. A stream contains a sequence of bytes that can be read from or written to by an application, and they can exist only in storages.

Table object: An object that is used to view properties for a collection of objects of a specific type, such as a **Message object** or a **Folder object**. A Table object is structured in a row and column format with each row representing an object and each column representing a property of the object.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

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

[MS-OXCFXICS] Microsoft Corporation, "Bulk Data Transfer Protocol".

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

[MS-OXCNOTIF] Microsoft Corporation, "Core Notifications Protocol".

[MS-OXCPERM] Microsoft Corporation, "Exchange Access and Operation Permissions Protocol".

[MS-OXCPRPT] Microsoft Corporation, "Property and Stream Object Protocol".

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

[MS-OXCSTOR] Microsoft Corporation, "Store Object Protocol".

[MS-OXCTABL] Microsoft Corporation, "Table Object Protocol".

[MS-OXORULE] Microsoft Corporation, "Email Rules Protocol".

[MS-OXOSFLD] Microsoft Corporation, "Special Folders Protocol".

[MS-OXOSRCH] Microsoft Corporation, "Search Folder List Configuration Protocol".

[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-OXCRPC] Microsoft Corporation, "Wire Format Protocol".

[MS-OXOMSG] Microsoft Corporation, "Email Object Protocol".

[MS-OXPROTO] Microsoft Corporation, "Exchange Server Protocols System Overview".

1.3 Overview

A folder is the basic unit of organization for **messaging objects** in a **message store**. A folder is represented in the message 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 message 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 message 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 Store Object Protocol, described in [MS-OXCSTOR], to log on to the message 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].

For conceptual background information and overviews of the relationships and interactions between this and other protocols, see [MS-OXPROTO].

1.5 Prerequisites/Preconditions

This protocol assumes that the client has previously logged on to the message 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 message store or to operate on an existing folder in a message 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

The **ROP request buffers** and **ROP response buffers** specified by this protocol are sent to and received by the server by using the underlying Remote Operations (ROP) List and Encoding Protocol, as specified in [MS-OXCROPS].

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**, 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 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].

2.2.1.1.1 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.<1>

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-OXCDATA] section 2.4.

HasRules (1 byte): A Boolean value that indicates whether **rules** are associated with the folder. If rules are associated with the folder, this field is set to a nonzero (TRUE) value; otherwise, this field is set to zero (FALSE).<2> For details about rules, see [MS-OXORULE].

IsGhosted (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 message 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 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 **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.

2.2.1.2 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 **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 2.2.2.2.2.5).

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-OXCDATA] section 2.4.

FolderId (8 bytes): A **Folder ID (FID)** structure ([MS-OXCDATA] 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 public folder with the name given by the **DisplayName** field of the request buffer (section 2.2.1.2.1) already exists.<3> The value is zero (FALSE) if a public folder with that name does not exist. This field applies only to a folder that is created in a public message store. The server always sets this field to zero for a folder that is created in a private mailbox.

HasRules (1 byte): A Boolean value that indicates whether rules are associated with the folder. If rules are associated with the folder, this field is set to a nonzero (TRUE) value; otherwise, this field is set to zero (FALSE). For details about rules, 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 message 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 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].

2.2.1.3.1 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 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, 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. If neither bit is set, the default is RESTART_SEARCH.
RESTART_SEARCH	0x00000002	The search is initiated, if this is the first **RopSetSearchCriteria** ROP request, or restarted, if the search is inactive. This bit MUST NOT be set at the same

Bit name	Value	Meaning
		time as the STOP_SEARCH bit. If neither bit is set, the default is RESTART_SEARCH.
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 bit is set, the default is 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 bit is set, the default is SHALLOW_SEARCH.
CONTENT_INDEXED_SEARCH	0x00010000	The search uses a content-indexed search. This bit MUST NOT be set at the same time as the NON_CONTENT_INDEXED_SEARCH bit. If neither bit is set, the default is at the discretion of the server. For more details, see section 3.2.5.4.
NON_CONTENT_INDEXED_SEARCH	0x00020000	The search does not use a content-indexed search. This bit MUST NOT be set at the same time as the CONTENT_INDEXED_SEARCH bit. If neither bit is set, the default is at the discretion of the server. For more details, see section 3.2.5.4.
STATIC_SEARCH	0x00040000	If set, the search is static. If not set, the search is dynamic.

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 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 for the search folder (2). For details about the structures that are used to specify restrictions, 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_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 without using content indexing.

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. The source folder can be a search folder (2), but the destination folder cannot. 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. This folder can be a search folder (2).

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. This folder cannot be a search folder (2).

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-OXCDATA] 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-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).

2.2.1.7 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-OXCDATA] 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-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).

2.2.1.8 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.<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].

2.2.1.8.1 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-OXCDATA] 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-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).

2.2.1.9 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). This ROP applies to both public folders and private mailboxes.<5>

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-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).

2.2.1.10 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. This ROP applies to both public folders and private mailboxes.<6>

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-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).

2.2.1.11 RopDeleteMessages ROP

The **RopDeleteMessages** ROP ([MS-OXCROPS] section 2.2.4.11) is used to soft delete one or more messages from a folder. This ROP applies to both public folders and private mailboxes.

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 and saving a Message object, see [MS-OXCMSG]. For details about copying a Message object from one folder to another, see section 3.1.4.7.

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. 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].

2.2.1.12.1 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. The folder can be either a public folder or a private mailbox 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 folders that are soft deleted. If this bit is not set, the hierarchy table lists only the existing folders. The listing of soft-deleted folders in a folder's hierarchy table depends on whether the soft-deleted folders remain part of that folder's hierarchy. The location where soft-deleted folders are stored is up to the implementer of the protocol.
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. This ROP applies to both public folders and private mailboxes.

Note: This ROP returns a Table object on which table operations can be performed. For example, the table's columns, which represent properties, can be set by using the **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1). 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<7> 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 messages that are soft deleted. If this bit is not set, the contents table lists only the existing messages.

Bit name	Value	Meaning
		The listing of soft-deleted messages in a folder's contents table depends on whether the soft-deleted messages remain part of that folder's contents. The location where soft-deleted messages are stored is up to the implementer of the protocol.
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 2.2.2.2.2.9. 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 specified as read-only, it means that the property can be set only by the server and clients SHOULD NOT try to change the value of this property. For details about how the server handles a client's attempt to set a read-only property, see [MS-OXCPRPT] section 3.2.5.4.

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 2.637) 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 2.639) 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 2.661) 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 2.512) contains an **Address Book EntryID** structure, as specified in [MS-OXCDATA] section 2.2.5.2, that specifies the name-service **entry ID** of a directory object that refers to a public folder. This property is set only for public folders.<8> 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 2.691) contains a **FID** structure ([MS-OXCDATA] section 2.2.1.1) that uniquely identifies a folder.

2.2.2.2.1.6 PidTagParentEntryId Property

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

The **PidTagParentEntryId** property ([MS-OXPROPS] section 2.847) contains a **Folder EntryID** structure, as specified in [MS-OXCDATA] section 2.2.4.1, that specifies the entry ID of the folder that contains the message or subfolder.

2.2.2.2.1.7 PidTagHierarchyChangeNumber Property

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

The **PidTagHierarchyChangeNumber** property ([MS-OXPROPS] section 2.710) 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.

2.2.2.2.1.8 PidTagMessageSize Property

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

The **PidTagMessageSize** property ([MS-OXPROPS] section 2.785) specifies the aggregate size of messages in the folder.

2.2.2.2.1.9 PidTagMessageSizeExtended Property

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

The **PidTagMessageSizeExtended** property ([MS-OXPROPS] section 2.786) specifies the 64-bit version of the **PidTagMessageSize** property (section 2.2.2.2.1.8).

2.2.2.2.1.10 PidTagSubfolders Property

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

The **PidTagSubfolders** property ([MS-OXPROPS] section 2.1020) specifies whether the folder has any subfolders. The value of this property is nonzero if the folder has subfolders; the value is zero otherwise.

2.2.2.2.1.11 PidTagLocalCommitTime Property

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

The **PidTagLocalCommitTime** property ([MS-OXPROPS] section 2.761) specifies the time, in **Coordinated Universal Time (UTC)**, that the folder was last changed.

2.2.2.2.1.12 PidTagLocalCommitTimeMax Property

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

The **PidTagLocalCommitTimeMax** property ([MS-OXPROPS] section 2.762) specifies the most recent time that a top level object within a folder was changed. Top level objects include messages and subfolders within the folder but not objects within the folder's subfolders.

2.2.2.2.1.13 PidTagDeletedCountTotal Property

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

The **PidTagDeletedCountTotal** property ([MS-OXPROPS] section 2.660) specifies the total number of messages that have been deleted from a folder, excluding messages that have been deleted from the folder's subfolders.

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 2.602) 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 2.628) 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 **RopCreateFolder** request (section 2.2.2.1).

2.2.2.2.2.3 PidTagContainerClass Property

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

The **PidTagContainerClass** property ([MS-OXPROPS] section 2.633) specifies the type of Message object that the folder contains. The value of this property MUST begin with "IPF.".

2.2.2.2.2.4 PidTagContainerHierarchy Property

Type: **PtypObject** ([MS-OXCDATA] section 2.11.1)

The **PidTagContainerHierarchy** property ([MS-OXPROPS] section 2.636) contains identifiers of the subfolders that are contained in the folder. This property is used in a download operation, as specified in [MS-OXCFXICS] section 2.2.1.7 and section 3.2.5.10.

2.2.2.2.2.5 PidTagDisplayName Property

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

The **PidTagDisplayName** property ([MS-OXPROPS] section 2.667) specifies the display name of the folder. Sibling folders MUST have unique display names.

2.2.2.2.2.6 PidTagFolderAssociatedContents Property

Type: **PtypObject** ([MS-OXCDATA] section 2.11.1)

The **PidTagFolderAssociatedContents** property ([MS-OXPROPS] section 2.690) contains identifiers of the FAI messages that are contained in the folder. This property is used in a download operation, as specified in [MS-OXCFXICS] section 2.2.1.7 and section 3.2.5.10.

2.2.2.2.2.7 PidTagFolderType Property

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

The **PidTagFolderType** property ([MS-OXPROPS] section 2.692) 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.

2.2.2.2.2.8 PidTagRights Property

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

The **PidTagRights** property ([MS-OXPROPS] section 2.924) specifies the client's folder permissions. This property's format is the same as that of the **PidTagMemberRights** property ([MS-OXCPERM] section 2.2.7). The FreeBusyDetailed flag and the FreeBusySimple flag do not apply to the **PidTagRights** property.

2.2.2.2.2.9 PidTagAccessControlListData Property

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

The **PidTagAccessControlListData** property ([MS-OXCPERM] section 2.2.3) contains the **access control list (ACL)** of the folder.<9>

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 subfolders contained in a folder. Each row of a hierarchy table contains a set of columns with information about one folder. A hierarchy table is implemented by message store providers and is used primarily by clients to show the hierarchy within a folder, displayed as 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 is 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 the Message objects contained in a folder. Each row of a contents table contains a set of columns with information about one Message object. A contents table is implemented by message store providers and is used primarily by clients and to show the messages contained in a folder.

The following are the four types of contents tables:

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

A standard contents table contains only standard (non-FAI) messages. A **FAI contents table** contains 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 2.2.2.2.1.5) 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 to specify the search criteria and FIDs ([MS-OXCDATA] section 2.2.1.1) to specify the search scope. A search folder (2) cannot be included in its own search scope. Therefore, the **FolderIds** field MUST NOT include the FID of the search folder (2).

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-OXCDATA] 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-OXCDATA] 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 and saving a Message object, see [MS-OXCMSG]. For details about copying a Message object from one folder to another, see section 3.1.4.7.

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<10> and the client has sufficient access rights to view the folder.

The server opens the folder according to the setting of the OpenSoftDeleted bit of the **OpenModeFlags** field of the ROP request buffer. 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-OXCDATA] section 2.4.

Error code name	Value	Meaning
ecNotFound	0x8004010F	The FID ([MS-OXCDATA] 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.<11>

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 **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 message store<12>.
ecNotFound	0x8004010F	The ROP was called on a Folder object that is a soft delete 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.

If the client sets an invalid bit in the **DeleteFolderFlags** field of the ROP request buffer, the server SHOULD<13> fail the ROP with an ecInvalidParam (0x80070057) error.

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.
ecInvalidParam	0x80070057	An invalid value was specified in a field.
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 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 and can be expensive. In the worst case, a search that is not content-indexed can involve the entire contents of the mailbox while holding an exclusive lock on the mailbox. A content-indexed search MAY<14> be static regardless of the value of the STATIC_SEARCH bit in the **RopSetSearchCriteria** ROP request.

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, which includes nontext **property types** such as date and time. Does not use a full-text search.

When the server receives a subsequent **RopSetSearchCriteria** ROP request in which neither STOP_SEARCH nor RESTART_SEARCH is set or in which neither RECURSIVE_SEARCH nor SHALLOW_SEARCH is set, the server SHOULD<15> use the default values.

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 the client does not specify FIDs, as specified in ([MS-OXCDATA] section 2.2.1.1), in the initial **RopSetSearchCriteria** ROP request, the server fails the ROP with ecNotInitialized (0x80040605). If the client does not specify FIDs in a subsequent **RopSetSearchCriteria** ROP request, the server uses the FIDs that were specified in the previous request. If the client sets the search scope to include the search folder (2) itself, the server SHOULD<16> fail the ROP with ecSearchFolderScopeViolation (0x00000490). If the client sets an invalid bit in the **SearchFlags** field, the server SHOULD<17> 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 were specified for this search folder (2).
ecNotSearchFolder	0x00000461	The object is not a search folder (2).
ecSearchFolderScopeViolation	0x00000490	The search folder (2) was included in its own search scope.
ecTooComplex	0x80040117	The restriction is too complex. The definition of complexity is

Error code name	Value	Meaning
		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.
ecSearchFolder	0x00000460	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.

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.

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. For details about how a message is opened with read/write access, see [MS-OXCMSG] section 3.1.4.1. 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.

If the client attempts to empty either the Root folder or a search folder, the server SHOULD<18> return ecNotSupported (0x80040102) in the **ReturnValue** field of the **RopEmptyFolder** ROP response buffer.

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	This ROP was called on a folder that is not allowed to be emptied or on an object that 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 **RopDelete**Messages 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. 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<19> 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. In the contents table for a search folder (2), the **PidTagParentEntryId** property (section 2.2.2.2.1.6) 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<20> 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)<21>. If the client sets an invalid bit in the **TableFlags** field, the server SHOULD<22> 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 and obfuscated, as described in [MS-OXCRPC] section 3. 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.

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 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-OXCDATA] 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 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, 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 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 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 block is 297 bytes.

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

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 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.")

AND (**PidTagImportance** is equal to 0x00000002)

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 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, 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 data in the response is expected to be in Unicode strings.

IncludeRestriction: 0x01 (TRUE). The server response is expected to include the restriction 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 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 block, which is 297 bytes.

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

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 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 Exchange Server 2013
- Microsoft Exchange Server 2016 Preview
- Microsoft Office Outlook 2003
- Microsoft Office Outlook 2007
- Microsoft Outlook 2010
- Microsoft Outlook 2013
- Microsoft Outlook 2016 Preview

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

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

<1> Section 2.2.1.1.1: Exchange 2013 and Exchange 2016 Preview ignore the OpenSoftDeleted bit and always opens an existing folder. Exchange 2013 and Exchange 2016 Preview do not provide access to soft-deleted folders.

<2> Section 2.2.1.1.2: Exchange 2003 and Exchange 2007 return zero (FALSE) in the **HasRules** field, even when there are rules on the **Inbox folder**.

<3> Section 2.2.1.2.2: Exchange 2010, Exchange 2013, and Exchange 2016 Preview always return zero (FALSE) in the **IsExistingFolder** field regardless of the existence of the named public folder.

<4> Section 2.2.1.8: Exchange 2013 and Exchange 2016 Preview do not support the **RopCopyFolder** ROP ([MS-OXCROPS] section 2.2.4.8) for public folders.

<5> Section 2.2.1.9: Exchange 2013 and Exchange 2016 Preview do not support the **RopEmptyFolder** ROP ([MS-OXCROPS] section 2.2.4.9) for public folders.

<6> Section 2.2.1.10: Exchange 2013 and Exchange 2016 Preview do not support the **RopHardDeleteMessagesAndSubfolders** ROP ([MS-OXCROPS] section 2.2.4.10) for public folders.

<7> Section 2.2.1.14.1: Exchange 2003 and Exchange 2007 do not support the ConversationMembers bit.

<8> Section 2.2.2.2.1.4: Exchange 2013 and Exchange 2016 Preview do not support the **PidTagAddressBookEntryId** property.

<9> Section 2.2.2.2.2.9: Outlook 2010, Outlook 2013, and Outlook 2016 Preview do not use the **PidTagAccessControlListData** property ([MS-OXCPERM] section 2.2.3).

<10> Section 3.2.5.1: If the specified folder has been hard deleted, Exchange 2007 does not fail the **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1), but no folder is opened.

<11> Section 3.2.5.2: Exchange 2007, Exchange 2013, and Exchange 2016 Preview return ecAccessdenied. Exchange 2010 returns ecNoCreateSubfolderRight.

<12> Section 3.2.5.2: Exchange 2010 and Exchange 2007 return ecError. Exchange 2013 and Exchange 2016 Preview return ecNotSupported.

<13> Section 3.2.5.3: Exchange 2010, Exchange 2013, and Exchange 2016 Preview ignore invalid bits instead of failing the ROP.

<14> Section 3.2.5.4: A content-indexed search is always static on the initial release version of Exchange 2010 and Exchange 2007 regardless of the value of the STATIC_SEARCH bit in the **RopSetSearchCriteria** request.

<15> Section 3.2.5.4: Exchange 2003, Exchange 2007, and Exchange 2010 preserve the values of the STOP_SEARCH, RESTART_SEARCH, RECURSIVE_SEARCH, and SHALLOW_SEARCH bits from the previous **RopSetSearchCriteria** ROP request and will use the preserved values instead of the default values.

<16> Section 3.2.5.4: Exchange 2003, Exchange 2007, and Exchange 2010 do not fail the **RopSetSearchCriteria** ROP when the search folder (2) is included in its own search scope.

<17> Section 3.2.5.4: Exchange 2007 silently ignores invalid bits and does not return the ecInvalidParam error code.

<18> Section 3.2.5.9: Exchange 2003, Exchange 2007, the initial release version of Exchange 2010, and Microsoft Exchange Server 2010 Service Pack 1 (SP1) do not return ecNotSupported (0x80040102) when the **RopEmptyFolder** ROP ([MS-OXCROPS] section 2.2.4.9) is called on the Root folder. Update Rollup 4 for Exchange Server 2010 Service Pack 2 (SP2), Exchange 2013, and Exchange 2016 Preview return ecNotSupported when the **RopEmptyFolder** ROP is called on the Root folder.

<19> Section 3.2.5.13: Exchange 2007 ignores invalid bits instead of failing the ROP.

<20> Section 3.2.5.14: Exchange 2003 fails the ROP with an error code of ecInvalidParam. Exchange 2007 ignores the ConversationMembers bit.

<21> Section 3.2.5.14: Exchange 2013 and Exchange 2016 Preview do not fail the operation and allow the setting of this combination of TableFlags.

<22> 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 this document since the last release. 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.
- The removal of a document from the documentation set.

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 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 changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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.
- 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 dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2 Messages	Updated product behavior notes for the "Messages" section to include behavior of Exchange 2016 and Outlook 2016.	Y	Product behavior note updated.
3 Protocol Details	Updated product behavior notes for the "Protocol Details" section to include behavior of Exchange 2016.	Y	Product behavior note updated.
6 Appendix A: Product Behavior	Added Exchange 2016 and Outlook 2016 to the list of applicable products.	Y	Content update.

8 Index

T

V