[MS-OXCPRPT]:

Property and Stream Object Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation 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 might 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, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial Availability.
4/25/2008	0.2	Minor	Revised and updated property names and other technical content.
6/27/2008	1.0	Major	Initial Release.
8/6/2008	1.01	Minor	Revised and edited technical content.
9/3/2008	1.02	Minor	Updated references.
12/3/2008	1.03	Minor	Updated IP notice
3/4/2009	1.04	Minor	Revised and edited technical content.
4/10/2009	2.0	Major	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	4.1.0	Minor	Updated the technical content.
5/5/2010	4.1.1	Editorial	Revised and edited the technical content.
8/4/2010	4.2	Minor	Clarified the meaning of the technical content.
11/3/2010	5.0	Major	Significantly changed the technical content.
3/18/2011	5.0	None	No changes to the meaning, language, and formatting of the technical content.
8/5/2011	5.1	Minor	Clarified the meaning of the technical content.
10/7/2011	6.0	Major	Significantly changed the technical content.
1/20/2012	7.0	Major	Significantly changed the technical content.
4/27/2012	7.1	Minor	Clarified the meaning of the technical content.
7/16/2012	7.1	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	8.0	Major	Significantly changed the technical content.
2/11/2013	9.0	Major	Significantly changed the technical content.
7/26/2013	9.1	Minor	Clarified the meaning of the technical content.
11/18/2013	9.2	Minor	Clarified the meaning of the technical content.
2/10/2014	9.2	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	9.2	None	No changes to the meaning, language, or formatting of the technical content.
7/31/2014	9.2	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
10/30/2014	10.0	Major	Significantly changed the technical content.
3/16/2015	11.0	Major	Significantly changed the technical content.
5/26/2015	11.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2015	11.0	None	No changes to the meaning, language, or formatting of the technical content.
6/13/2016	11.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	11.0	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Property and Stream Object Protocol enables a client to read, set, and delete the properties of an object.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

address book container: An **Address Book object** that describes an address list.

Address Book object: An entity in an address book that contains a set of attributes, each attribute with a set of associated values.

Attachment object: A set of properties that represents a file, **Message object**, or structured storage that is attached to a Message object and is visible through the **attachments table** for a Message object.

attachments table: A Table object whose rows represent the **Attachment objects** that are attached to a **Message object**.

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 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**.

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).

distribution list: A collection of users, computers, contacts, or other groups that is used only for email distribution, and addressed as a single recipient.

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.

global identifier: A form of encoding for an internal identifier that makes it unique across all stores. Global identifiers are a subset of external identifiers, and they consist of a REPLGUID followed by a 6-byte global counter.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also universally unique identifier (UUID).

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

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

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.

long ID (LID): A 32-bit quantity that, in combination with a GUID, defines a **named property**.

mail user: An **Address Book object** that represents a person or entity that can receive deliverable messages.

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.

multibyte character set (MBCS): An alternative to **Unicode** for supporting character sets, like Japanese and Chinese, that cannot be represented in a single byte. Under MBCS, characters are encoded in either one or two bytes. In two-byte characters, the first byte, or "lead" byte, signals that both it and the following byte are to be interpreted as one character. The first byte comes from a range of codes reserved for use as lead bytes. Which ranges of bytes can be lead bytes depends on the **code page** in use. For example, Japanese **code page** 932 uses the range 0x81 through 0x9F as lead bytes, but Korean **code page** 949 uses a different range.

named property: A property that is identified by both a GUID and either a string name or a 32-bit identifier.

property ID: A 16-bit numeric identifier of a specific attribute (1). A property ID does not include any **property type** information.

property name: A string that, in combination with a **property set**, identifies a **named property**.

property set: A set of attributes (1), identified by a **GUID**. Granting access to a property set grants access to all the attributes in the set.

property tag: A 32-bit value that contains a property type and a property ID. The low-order 16 bits represent the property type. The high-order 16 bits represent the property ID.

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.

recipient table: The part of a **Message object** that represents users to whom a message is addressed. Each row of the table is a set of properties that represents one recipient (2).

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.

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.

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**.

Store object: An object that is used to store mailboxes and **public folder** content.

Stream object: A **Server object** that is used to read and write large string and binary properties.

string property: A property whose property type is PtypString8 or PtypString.

tagged property: A property that is defined by a 16-bit property ID and a 16-bit property type. The property ID for a tagged property is in the range 0x001 – 0x7FFF. Property IDs in the range 0x8000 – 0x8FFF are reserved for assignment to **named properties**.

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-OXCFOLD] Microsoft Corporation, "Folder Object Protocol".

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

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

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

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

1.3 Overview

This protocol enables a client to access and manage the properties of a **Server object** by using **remote operations (ROPs)**. The properties store data about the object.

Properties can be set, retrieved, and deleted. Properties can also be copied from one object to another. A client can copy just a select few properties or copy all properties. Some properties on **Message objects** and **Attachment objects** can be opened as a stream. With an open stream, the client can seek, read, write, and commit data to the stream.

All operations on a property require that the property is specified by a **property tag**, which comprises the **property ID** and the **property type**. A property is either a **tagged property**, which has a hard-coded property ID, or a **named property**, which has a dynamic property ID assigned by the server. A client can create custom properties by defining its own named properties. A client can obtain a property ID for a named property by querying the server for the named property-to-property ID mapping. A query for the reverse mapping (property ID-to-named property) allows the client to obtain the named property from the property ID.

There are two different models for saving changes to properties. Changes to properties on **Folder objects** and **Logon objects** are saved immediately, whereas changes to properties on Message objects and Attachment objects are not saved until the client explicitly saves the object.

1.4 Relationship to Other Protocols

The Property and Stream 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**.

Any protocol that reads, sets, or deletes the properties of an object relies on the Property and Stream Object Protocol.

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 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. Methods to open the object and acquire a handle are dependent on the type of object. For details about Message objects and Attachment objects, see [MS-OXCMSG]. For details about Folder objects, see [MS-OXCFOLD]. For details about Logon objects, see [MS-OXCSTOR]. Details about **Stream objects** are specified in this document.

1.6 Applicability Statement

A client can use this protocol to perform all operations on the properties of a Message object, an Attachment object, a Folder object, or a Logon object.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

A client can create its own custom property for an object by defining a named property. A named property has a dynamic property ID that is assigned by the server. A mapping of named properties to property IDs is provided by the server.

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 Common Object Properties

The properties specified in section 2.2.1.1 through section 2.2.1.9 are present on all objects. A property that is specified as "read-only for the client" can be set only by the server.

2.2.1.1 PidTagAccess Property

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

The **PidTagAccess** property ([MS-OXPROPS] section 2.496) indicates the operations available to the client for the object. The value is a bitwise OR of zero or more values from the following table. This property is read-only for the client.

Value	Meaning
0x00000001	Modify
0x00000002	Read
0x00000004	Delete
0x00000008	Create **hierarchy table**
0x00000010	Create **contents table**
0x00000020	Create associated contents table

2.2.1.2 PidTagAccessLevel Property

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

The **PidTagAccessLevel** property ([MS-OXPROPS] section 2.498) indicates the client's access level to the object. This property does not apply to Folder objects and Logon objects. This value of this property MUST be one of the values in the following table. This property is read-only for the client.

Value	Meaning
0x00000000	Read-only
0x00000001	Modify

2.2.1.3 PidTagChangeKey Property

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

The **PidTagChangeKey** property ([MS-OXCFXICS] section 2.2.1.2.7) contains a **global identifier** indicating the last change to the object. This property is read-only for clients.

2.2.1.4 PidTagCreationTime Property

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

The **PidTagCreationTime** property ([MS-OXCMSG] section 2.2.2.3) contains the time that the object was created in **Coordinated Universal Time (UTC)**. This property is read-only for clients.

2.2.1.5 PidTagLastModifierName Property

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

The **PidTagLastModifierName** property ([MS-OXPROPS] section 2.757) contains the name of the last **mail user** to modify the object. This property does not apply to Folder objects and Logon objects. This property is read-only for clients.

2.2.1.6 PidTagLastModificationTime Property

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

The **PidTagLastModificationTime** property ([MS-OXCMSG] section 2.2.2.2) contains the time of the last modification to the object in UTC. This property is read-only for clients.

2.2.1.7 PidTagObjectType Property

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

The **PidTagObjectType** property ([MS-OXPROPS] section 2.804) indicates the type of Server object. This property does not apply to Folder objects and Logon objects. The value of this property MUST be one of the values in the following table. This property is read-only for the client.

Value	Meaning
0x00000001	**Store object**
0x00000002	**Address Book object**
0x00000004	**address book container**
0x00000005	Message object
0x00000006	mail user
0x00000007	Attachment object
0x00000008	**distribution list**

2.2.1.8 PidTagRecordKey Property

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

The **PidTagRecordKey** property ([MS-OXPROPS] section 2.901) contains a unique binary-comparable identifier for a specific object. Whenever a copy of an object is created, the server generates a new identifier for the copied object. This property does not apply to Folder objects and Logon objects. This property is read-only for the client.

2.2.1.9 PidTagSearchKey Property

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

The **PidTagSearchKey** property ([MS-OXPROPS] section 2.988) contains a unique binary-comparable key that identifies an object for a search. Whenever a copy of an object is created, the key is also copied from the original object. This property does not apply to Folder objects and Logon objects. This property is read-only for clients.

2.2.2 RopGetPropertiesSpecific ROP

The **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3) queries for and returns the values of properties specified in the **PropertyTags** field. This operation is valid on Message objects, Folder objects, Attachment objects and Logon objects.

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].

2.2.2.1 RopGetPropertiesSpecific ROP Request Buffer

The following descriptions define valid fields for the **RopGetPropertiesSpecific** ROP request buffer ([MS-OXCROPS] section 2.2.8.3.1).

PropertySizeLimit (2 bytes): An integer that specifies the maximum size allowed for a property value. If this value is zero, the property values are limited only by the size of the ROP response buffer. If this value is nonzero, the property values are limited both by the size of the ROP response buffer and by the value of the **PropertySizeLimit** field.

WantUnicode (2 bytes): A Boolean value that is nonzero if **string properties** that are requested with **PtypUnspecified** ([MS-OXCDATA] section 2.11.1) as the property type are to be encoded in the **Unicode** format in the ROP response buffer. If **WantUnicode** is set to zero, the string properties that are requested with **PtypUnspecified** as the property type are to be encoded in **multibyte character set (MBCS)** format.

PropertyTagCount (2 bytes): An integer that specifies the number of property tags contained in the **PropertyTags** field.

PropertyTags (variable): An array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). Each structure contains the property tag of a property for which the client is requesting the value.

2.2.2.2 RopGetPropertiesSpecific ROP Response Buffer

The following descriptions define valid fields for the **RopGetPropertiesSpecific** ROP response buffer ([MS-OXCROPS] section 2.2.8.3.2).

RowData (variable): A **PropertyRow** structure ([MS-OXCDATA] section 2.8.1) that contains the values of the properties specified in the ROP request buffer.

2.2.3 RopGetPropertiesAll ROP

The **RopGetPropertiesAll** ROP ([MS-OXCROPS] section 2.2.8.4) queries for and returns all of the property tags and values of properties that have been set. The client can create an equivalent duplicate of a Message object by copying only these properties, without considering its **attachments table** and **recipient table** that might be on the object. This operation is valid on Message objects, Folder objects, Attachment objects, and Logon objects.

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].

2.2.3.1 RopGetPropertiesAll ROP Request Buffer

The following descriptions define valid fields for the **RopGetPropertiesAll** ROP request buffer ([MS-OXCROPS] section 2.2.8.4.1).

PropertySizeLimit (2 bytes): An integer that specifies the maximum size allowed for a property value. If this value is zero, the property values are limited only by the size of the ROP response buffer. If this value is nonzero, the property values are limited both by the size of the ROP response buffer and by the value of the **PropertySizeLimit** field.

WantUnicode (2 bytes): A Boolean value that is nonzero if string properties that are requested with **PtypUnspecified** ([MS-OXCDATA] section 2.11.1) as the property type are to be encoded in the Unicode format in the ROP response buffer. If **WantUnicode** is set to zero, the string properties that are requested with **PtypUnspecified** as the property type are to be encoded in MBCS format.

2.2.3.2 RopGetPropertiesAll ROP Response Buffer

The following descriptions define valid fields for the **RopGetPropertiesAll** ROP response buffer ([MS-OXCROPS] section 2.2.8.4.2).

PropertyValueCount (2 bytes): An integer that specifies the number of elements in the **PropertyValues** field.

PropertyValues (variable): An array of **TaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.4) containing all property tags and property values on the queried object. If a property value is larger than the available space in the ROP response buffer or if the property value is larger than the size specified in the **PropertySizeLimit** field of the ROP request buffer, the type MUST be **PtypErrorCode** ([MS-OXCDATA] section 2.11.1) with a value of **NotEnoughMemory** ([MS-OXCDATA] section 2.4.2).

2.2.4 RopGetPropertiesList ROP

The **RopGetPropertiesList** ROP ([MS-OXCROPS] section 2.2.8.5) queries for and returns all of the property tags for properties that have been set on an object. This operation is valid on Message objects, Folder objects, Attachment objects, and Logon objects.

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].

2.2.4.1 RopGetPropertiesList ROP Request Buffer

This protocol adds no additional field details to the **RopGetPropertiesList** ROP request buffer ([MS-OXCROPS] section 2.2.8.5.1).

2.2.4.2 RopGetPropertiesList ROP Response Buffer

The following descriptions define valid fields for the **RopGetPropertiesList** ROP response buffer ([MS-OXCROPS] section 2.2.8.5.2).

PropertyTagCount (2 bytes): An integer that specifies the number of property tags contained in the **PropertyTags** field.

PropertyTags (variable): An array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). The array contains a property tag for each property currently set on the object.

2.2.5 RopSetProperties ROP

The **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) updates the specified properties on an object. This operation is valid on Message objects, Folder objects, Attachment objects, and Logon objects.

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].

2.2.5.1 RopSetProperties ROP Request Buffer

The following descriptions define valid fields for the **RopSetProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.6.1).

PropertyValueSize (2 bytes): An integer that specifies the number of bytes in the **PropertyValueCount** field plus the number of bytes in the **PropertyValues** field.

PropertyValueCount (2 bytes): An integer that specifies the number of elements contained in the **PropertyValues** field.

PropertyValues (variable): An array of **TaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.4). Each structure specifies a property that is to be updated.

2.2.5.2 RopSetProperties ROP Response Buffer

The following descriptions define valid fields for the **RopSetProperties** ROP response buffer ([MS-OXCROPS] section 2.2.8.6.2).

PropertyProblemCount (2 bytes): An integer that specifies the number of elements contained in the **PropertyProblems** field.

PropertyProblems (variable): An array of **PropertyProblem** structures ([MS-OXCDATA] section 2.7). Each structure specifies a property that was not set and the reason that the property was not set.

2.2.6 RopSetPropertiesNoReplicate ROP

The **RopSetPropertiesNoReplicate** ROP ([MS-OXCROPS] section 2.2.8.7) has the same fields and works the same way as the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6), with the exception that when this ROP is used to set properties on a Folder object, the updated properties will not undergo folder replication. For information about folder replication, see [MS-OXCSTOR] section 3.1.4.3. On all other objects, the **RopSetPropertiesNoReplicate** ROP works the same way as the **RopSetProperties** ROP.

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].

2.2.6.1 RopSetPropertiesNoReplicate ROP Request Buffer

The following descriptions define valid fields for the **RopSetPropertiesNoReplicate** ROP request buffer ([MS-OXCROPS] section 2.2.8.7.1).

The fields of the **RopSetPropertiesNoReplicate** ROP request buffer ([MS-OXCROPS] section 2.2.8.7.1) are the same as those of the **RopSetProperties** ROP request buffer (section 2.2.5.1).

2.2.6.2 RopSetPropertiesNoReplicate ROP Response Buffer

The following descriptions define valid fields for the **RopSetPropertiesNoReplicate** ROP response buffer ([MS-OXCROPS] section 2.2.8.7.2).

The fields of the **RopSetPropertiesNoReplicate** ROP response buffer ([MS-OXCROPS] section 2.2.8.7.2) are the same as those of the **RopSetProperties** ROP response buffer (section 2.2.5.2).

2.2.7 RopDeleteProperties ROP

The **RopDeleteProperties** ROP ([MS-OXCROPS] section 2.2.8.8) removes the specified properties from an object. This operation is valid on Message objects, Folder objects, Attachment objects, and Logon objects.

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].

2.2.7.1 RopDeleteProperties ROP Request Buffer

The following descriptions define valid fields for the **RopDeleteProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.8.1).

PropertyTagCount (2 bytes): An integer that specifies the number of property tags contained in the **PropertyTags** field.

PropertyTags (variable): An array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). Each structure specifies the property tag of a property that the client is deleting.

2.2.7.2 RopDeleteProperties ROP Response Buffer

The following descriptions define valid fields for the **RopDeleteProperties** ROP response buffer ([MS-OXCROPS] section 2.2.8.8.2).

PropertyProblemCount (2 bytes): An integer that specifies the number of elements contained in the **PropertyProblems** field.

PropertyProblems (variable): An array of **PropertyProblem** structures ([MS-OXCDATA] section 2.7). Each structure specifies a property that was not deleted and the reason that the property was not deleted.

2.2.8 RopDeletePropertiesNoReplicate ROP

The **RopDeletePropertiesNoReplicate** ROP ([MS-OXCROPS] section 2.2.8.9) has the same fields and works the same way as the **RopDeleteProperties** ROP ([MS-OXCROPS] section 2.2.8.8), with

the exception that when this ROP is used to delete properties from a Folder object, the deleted properties will not undergo folder replication. For information about folder replication, see [MS-OXCSTOR] section 3.1.4.3. On all other objects, the **RopDeletePropertiesNoReplicate** ROP works the same way as **RopDeleteProperties**.

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].

2.2.8.1 RopDeletePropertiesNoReplicate ROP Request Buffer

The following descriptions define valid fields for the **RopDeletePropertiesNoReplicate** ROP request buffer ([MS-OXCROPS] section 2.2.8.9.1).

The fields of the **RopDeletePropertiesNoReplicate** ROP request buffer are the same as those of the **RopDeleteProperties** ROP request buffer (section 2.2.7.1).

2.2.8.2 RopDeletePropertiesNoReplicate ROP Response Buffer

The following descriptions define valid fields for the **RopDeletePropertiesNoReplicate** ROP response buffer ([MS-OXCROPS] section 2.2.8.9.2).

The fields of the **RopDeletePropertiesNoReplicate** ROP response buffer are the same as those of the **RopDeleteProperties** ROP response buffer (section 2.2.7.2).

2.2.9 RopQueryNamedProperties ROP

The **RopQueryNamedProperties** ROP ([MS-OXCROPS] section 2.2.8.10) queries an object for all the named properties. This operation is valid on Logon objects, Message objects, Folder objects, and Attachment objects.<1>

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].

2.2.9.1 RopQueryNamedProperties ROP Request Buffer

The following descriptions define valid fields for the **RopQueryNamedProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.10.1).

QueryFlags (1 byte): A set of bits that control which type of named property is returned. The valid bits for this field are listed in the following table.

Bit name	Value	Meaning
NoStrings	0x01	Named properties that have a **property name** identifier MUST NOT be included in the response.
NoIds	0x02	Named properties that have a **long ID (LID)** MUST NOT be included in the response.

HasGuid (1 byte): A Boolean value that is nonzero if the **PropertyGuid** field is included in the request. If the value is zero, the **PropertyGUID** field MUST NOT be present.

PropertyGuid (16 bytes): A **GUID** that specifies the **property set** of properties to be returned. If this field is present, only named properties with a property set matching the GUID are returned in a successful response. This field MUST be present if the **HasGuid** field is set to nonzero.

2.2.9.2 RopQueryNamedProperties ROP Response Buffer

The following descriptions define valid fields for the **RopQueryNamedProperties** ROP response buffer ([MS-OXCROPS] section 2.2.8.10.2).

IdCount (2 bytes): An integer that specifies the number of elements contained in the **PropertyIds** field.

PropertyIds (variable): An array of 16-bit integers, each of which is a property ID. The number of integers contained in the array MUST equal the value specified in the **IdCount** field. The array MUST contain one property ID for each of the named properties specified in the **PropertyNames** field.

PropertyNames (variable): An array of **PropertyName** structures ([MS-OXCDATA] section 2.6.1). Each structure contains the details about a named property. The entries in this list MUST match the order of the entries in the **PropertyIds** field, and the number of entries MUST be equal.

2.2.10 RopCopyProperties ROP

The **RopCopyProperties** ROP ([MS-OXCROPS] section 2.2.8.11) copies or moves one or more properties from one object to another. It can be used for replying to and forwarding Message objects in which only some of the properties from the original Message object travel with the reply or forwarded copy. Also, the source and destination object MUST be of the same type.This operation is valid on Folder objects, Attachment objects, and Message objects.

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].

2.2.10.1 RopCopyProperties ROP Request Buffer

The following descriptions define valid fields for the **RopCopyProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.11.1).

WantAsynchronous (1 byte): A Boolean value that indicates whether the ROP is to be processed synchronously or asynchronously. If this field is set to zero, the ROP is processed synchronously. If this field is set to nonzero, the ROP is processed either synchronously or asynchronously.

CopyFlags (1 byte): A set of bits that control options for moving or copying properties. The valid bits are listed in the following table. These bits SHOULD NOT<2> be combined.

Bit name	Value	Meaning
Move	0x01	If this bit is set, properties are moved; otherwise, properties are copied.
NoOverwrite	0x02	If this bit is set, properties that already have a value on the destination object will not be overwritten; otherwise, they are overwritten.

PropertyTagCount (2 bytes): An integer that specifies the number of elements contained in the **PropertyTags** field.

PropertyTags (variable): An array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). Each structure contains the property tag of a property to be copied or moved.

2.2.10.2 RopCopyProperties ROP Response Buffer

The following descriptions define valid fields for the **RopCopyProperties** ROP response buffer ([MS-OXCROPS] section 2.2.8.11.2).

PropertyProblemCount (2 bytes): An integer that specifies the number of elements contained in the **PropertyProblems** field. This field MUST NOT be present if the **ReturnValue** field is set to NullDestinationObject (0x00000503).

PropertyProblems (variable): An array of **PropertyProblem** structures ([MS-OXCDATA] section 2.7). Each structure specifies a property that was not copied or moved and the reason that the property was not copied or moved. This field MUST NOT be present if the **ReturnValue** field is set to NullDestinationObject (0x00000503).

DestHandleIndex (4 bytes): An integer that specifies the location in the **Server object handle table** where the handle for the destination Server object is stored. The **DestHandleIndex** field MUST be set to the value of the **DestHandleIndex** field of the ROP request buffer. The **DestHandleIndex** field MUST NOT be present if the **ReturnValue** field is set to any value other than NullDestinationObject (0x00000503).

2.2.11 RopCopyTo ROP

The **RopCopyTo** ROP ([MS-OXCROPS] section 2.2.8.12) is used to copy or move all but a specified few properties from a source object to a destination object. This operation is valid on Message objects, Attachment objects, and Folder objects.

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].

2.2.11.1 RopCopyTo ROP Request Buffer

The following descriptions define valid fields for the **RopCopyTo** ROP request buffer ([MS-OXCROPS] section 2.2.8.12.1).

WantAsynchronous (1 byte): A Boolean value that indicates whether the ROP is to be processed synchronously or asynchronously. If this field is set to zero, this ROP is processed synchronously. If this field is set to nonzero, this ROP is processed either synchronously or asynchronously.

WantSubObjects (1 byte): A Boolean value that is nonzero if subobjects are also to be copied. Otherwise, they are not.

CopyFlags (1 byte): A set of bits that control options for moving or copying properties. The valid bits are listed in the following table. These bits SHOULD NOT<3> be combined.

Bit name	Value	Meaning
Move	0x01	If this bit is set, properties are moved; otherwise, properties are copied.
NoOverwrite	0x02	If this bit is set, properties that already have a value on the destination object will not be overwritten; otherwise, they are overwritten.

ExcludedTagCount (2 bytes): An integer that specifies the number of elements contained in the **ExcludedTags** field.

ExcludedTags (variable): An array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). Each structure contains the property tag of a property that MUST NOT be copied or moved as part of this operation.

2.2.11.2 RopCopyTo ROP Response Buffer

The following descriptions define valid fields for the **RopCopyTo** ROP response buffer ([MS-OXCROPS] section 2.2.8.12.2).

PropertyProblemCount (2 bytes): An integer that specifies the number of elements contained in the **PropertyProblems** field. This field MUST NOT be present if the **ReturnValue** field is set to NullDestinationObject (0x00000503).

PropertyProblems (variable): An array of **PropertyProblem** structures ([MS-OXCDATA] section 2.7). Each structure specifies a property that was not copied or moved and the reason that the property was not copied or moved. This field MUST NOT be present if the **ReturnValue** field is set to NullDestinationObject (0x00000503).

DestHandleIndex (4 bytes): An integer that specifies the location in the Server object handle table where the handle for the destination Server object is stored. The **DestHandleIndex** field MUST be set to the value of the **DestHandleIndex** field of the ROP request buffer. The **DestHandleIndex** field MUST NOT be present if the **ReturnValue** field is set to any value other than NullDestinationObject (0x00000503).

2.2.12 RopGetPropertyIdsFromNames ROP

The **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1) maps abstract, client-defined named properties to concrete 16-bit property IDs (of which 15 bits are significant). This operation is valid on Message objects, Attachment objects, Folder objects, and Logon objects.

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].

2.2.12.1 RopGetPropertyIdsFromNames ROP Request Buffer

The following descriptions define valid fields for the **RopGetPropertyIdsFromNames** ROP request buffer ([MS-OXCROPS] section 2.2.8.1.1).

Flags (1 byte): An integer that specifies whether to create a new entry. This field is set to 0x02 to request that a new entry be created for each named property that is not found in the existing mapping table; this field is set to 0x00 otherwise.

PropertyNameCount (2 bytes): An integer that specifies the number of **PropertyName** structures contained in the **PropertyNames** field. A value of zero indicates that the client is querying the complete list of registered named properties.

PropertyNames (variable): An array of **PropertyName** structures ([MS-OXCDATA] section 2.6). Each structure specifies a named property to be mapped to a property ID.

2.2.12.2 RopGetPropertyIdsFromNames ROP Response Buffer

The following descriptions define valid fields for the **RopGetPropertyIdsFromNames** ROP response buffer ([MS-OXCROPS] section 2.2.8.1.2).

PropertyIdCount (2 bytes): An integer that specifies the number of property IDs contained in the **PropertyIds** field. The value of this field MUST be equal to the value of the **PropertyNameCount** field of the ROP request buffer unless the value of the **PropertyNameCount** field is zero. In that case, the value MUST be equal to the total number of registered named properties.

PropertyIds (variable): An array of 16-bit integers. Each integer is a property ID that is mapped from a named property that is specified in the **PropertyNames** field of the ROP request buffer. If a named property cannot be mapped, the associated entry in the **PropertyIds** field MUST be 0x0000. The order of property IDs in this array MUST match the order of the named properties specified in the **PropertyNames** field of the ROP request buffer.

Reasons a name couldn't be mapped include:

- Use of the PS_MAPI namespace and not specifying 0x00 for the **Kind** field of the **PropertyName** structure ([MS-OXCDATA] section 2.6.1).
- The name wasn't found in the mapping table and the **Flags** field of the ROP request buffer was not set to 0x02.
- The user does not have permission to register new named properties.
- The user has reached an artificial quota of named properties imposed by the server.
- The user has reached the hard limit of 32,767 registered named properties.

2.2.13 RopGetNamesFromPropertyIds ROP

The **RopGetNamesFromPropertyIds** ROP ([MS-OXCROPS] section 2.2.8.2) maps concrete property IDs to abstract, client-defined named properties. This operation is valid on Message objects, Attachment objects, Folder objects, and Logon objects. Property IDs for named properties are identified by having their most significant bit set (0x8000). The client can request to map non-named property IDs (IDs with the most significant bit unset) into names.

The PS_MAPI namespace is used for mapping non-named property IDs into names.

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].

2.2.13.1 RopGetNamesFromPropertyIds ROP Request Buffer

The following descriptions define valid fields for the **RopGetNamesFromPropertyIds** ROP request buffer ([MS-OXCROPS] section 2.2.8.2.1).

PropertyIdCount (2 bytes): An integer that specifies the number of 16-bit property IDs contained in the **PropertyIds** field.

PropertyIds (variable): An array of 16-bit integers. Each integer is a property ID to be mapped to a named property. The client can pass property ID values less than 0x8000. These values will be mapped into names in the PS_MAPI namespace ([MS-OXPROPS] section 1.3.2).

2.2.13.2 RopGetNamesFromPropertyIds ROP Response Buffer

The following descriptions define valid fields for the **RopGetNamesFromPropertyIds** ROP response buffer ([MS-OXCROPS] section 2.2.8.2.2).

PropertyNameCount (2 bytes): An integer that specifies the number of structures contained in the **PropertyNames** field. This value MUST be equal to the value of the **PropertyIdCount** field of the ROP request buffer.

PropertyNames (variable): An array of **PropertyName** structures ([MS-OXCDATA] section 2.6). Each structure specifies a named property that is mapped from a property ID that is specified in the **PropertyIds** field of the ROP request buffer.

2.2.14 RopOpenStream ROP

The **RopOpenStream** ROP ([MS-OXCROPS] section 2.2.9.1) opens a property as a Stream object, enabling the client to perform various streaming operations on the property. This operation is valid on Folder objects, Attachment objects, and Message objects. Only single-valued properties that are of the

following types are supported for Attachment objects and Message objects: **PtypBinary**, **PtypObject**, **PtypString8**, and **PtypString** ([MS-OXCDATA] section 2.11.1). Only single-valued properties that are of the following types MUST be supported for Folder objects: **PtypBinary**.

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].

2.2.14.1 RopOpenStream ROP Request Buffer

The following descriptions define valid fields for the **RopOpenStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.1.1).

PropertyTag (4 bytes): An integer that is the property tag of the property for which the client is opening the stream. The format of a property tag is specified in [MS-OXCDATA] section 2.9.

OpenModeFlags (1 byte): An integer that specifies the mode for opening the stream. The **OpenModeFlags** field MUST be set to one of the values in the following table.

Bit name	Value	Meaning
ReadOnly	0x00	Open the stream for read-only access.
ReadWrite	0x01	Open the stream for read/write access.
Create	0x02	Open a new stream. This mode will delete the current property value and open the stream for read/write access. This mode is required for a property that has not been set.

2.2.14.2 RopOpenStream ROP Response Buffer

The following descriptions define valid fields for the **RopOpenStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.1.2).

StreamSize (4 bytes): An integer that specifies the number of bytes in the opened stream.

2.2.15 RopReadStream ROP

The **RopReadStream** ROP ([MS-OXCROPS] section 2.2.9.2) reads the stream of bytes from a Stream object. This operation is valid only on Stream objects.

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].

2.2.15.1 RopReadStream ROP Request Buffer

The following descriptions define valid fields for the **RopReadStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.2.1).

ByteCount (2 bytes): An integer that specifies the maximum number of bytes to be read unless the value is 0xBABE. In that case, the maximum number of bytes to be read is specified by the **MaximumByteCount** field.

MaximumByteCount (4 bytes): An integer that specifies the maximum number of bytes to be read if the **ByteCount** field is set to 0xBABE. Note that because the value of the **MaximumByteCount** field can exceed the amount of data that can be returned in a single ROP response buffer, the

server's response can span multiple ROP response buffers. This field MUST be present if and only if the value of the **ByteCount** field is 0xBABE.

2.2.15.2 RopReadStream ROP Response Buffer

The following descriptions define valid fields for the **RopReadStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.2.2).

DataSize (2 bytes): An integer that specifies the number of bytes in the **Data** field.

Data (variable): An array of bytes that constitute the data read from the stream. This field MUST contain exactly the number of bytes specified in the **DataSize** field.

2.2.16 RopWriteStream ROP

The **RopWriteStream** ROP ([MS-OXCROPS] section 2.2.9.3) writes the stream of bytes into a Stream object. This operation is valid only on Stream objects.

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].

2.2.16.1 RopWriteStream ROP Request Buffer

The following descriptions define valid fields for the **RopWriteStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.3.1).

DataSize (2 bytes): An integer that specifies the number of bytes in the **Data** field.

Data (variable): An array of bytes that constitute the data to be written to the stream.

2.2.16.2 RopWriteStream ROP Response Buffer

The following descriptions define valid fields for the **RopWriteStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.3.2).

WrittenSize (2 bytes): An integer that specifies the number of bytes actually written to the stream.

2.2.17 RopCommitStream ROP

The **RopCommitStream** ROP ([MS-OXCROPS] section 2.2.9.4) ensures that any changes made to a Stream object are persisted in storage for a Folder object. This operation is valid only on Stream objects.

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].

2.2.17.1 RopCommitStream ROP Request Buffer

This protocol adds no additional field details to the **RopCommitStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.4.1).

2.2.17.2 RopCommitStream ROP Response Buffer

This protocol adds no additional field details to the **RopCommitStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.4.2).

2.2.18 RopGetStreamSize ROP

The **RopGetStreamSize** ROP ([MS-OXCROPS] section 2.2.9.5) retrieves the size of the stream. This operation is valid only on Stream objects.

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].

2.2.18.1 RopGetStreamSize ROP Request Buffer

This protocol adds no additional field details to the **RopGetStreamSize** ROP request buffer ([MS-OXCROPS] section 2.2.9.5.1).

2.2.18.2 RopGetStreamSize ROP Response Buffer

The following descriptions define valid fields for the **RopGetStreamSize** ROP response buffer ([MS-OXCROPS] section 2.2.9.5.2).

StreamSize (4 bytes): An integer that specifies the number of bytes in the stream. The maximum allowed stream size is 2^31 bytes.

2.2.19 RopSetStreamSize ROP

The **RopSetStreamSize** ROP ([MS-OXCROPS] section 2.2.9.6) sets the size of a stream. This operation is valid only on Stream objects. This ROP is not required prior to writing to the stream.

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].

2.2.19.1 RopSetStreamSize ROP Request Buffer

The following descriptions define valid fields for the **RopSetStreamSize** ROP request buffer ([MS-OXCROPS] section 2.2.9.6.1).

StreamSize (8 bytes): An integer that specifies the size, in bytes, of the stream. The maximum allowed stream size is 2^31 bytes.

2.2.19.2 RopSetStreamSize ROP Response Buffer

This protocol adds no additional field details to the **RopSetStreamSize** ROP response buffer ([MS-OXCROPS] section 2.2.9.6.2).

2.2.20 RopSeekStream ROP

The **RopSeekStream** ROP ([MS-OXCROPS] section 2.2.9.7) sets the seek pointer to a new location, which is relative to the beginning of the stream, the end of the stream, or the location of the current seek pointer. This ROP can also be used to get the location of the current seek pointer by setting the **Origin** field to 0x01 and the **Offset** field to zero. This operation is valid only on Stream objects.

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].

2.2.20.1 RopSeekStream ROP Request Buffer

The following descriptions define valid fields for the **RopSeekStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.7.1).

Origin (1 byte): An integer that specifies the point of origin for the seek pointer. The **Origin** field MUST be set to one of the values in the following table.

Value	Meaning
0x00	The point of origin is the beginning of the stream.
0x01	The point of origin is the location of the current seek pointer.
0x02	The point of origin is the end of the stream.

Offset (8 bytes): An integer that specifies the number of bytes that the seek pointer is to be offset from the origin. The value can be positive or negative.

2.2.20.2 RopSeekStream ROP Response Buffer

The following descriptions define valid fields for the **RopSeekStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.7.2).

NewPosition (8 bytes): An integer that specifies the new offset, in bytes, of the seek pointer from the beginning of the stream.

2.2.21 RopCopyToStream ROP

The **RopCopyToStream** ROP ([MS-OXCROPS] section 2.2.9.8) copies a specified number of bytes from the current seek pointer in the source stream to the current seek pointer in the destination stream. This operation is valid only on Stream objects.

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].

2.2.21.1 RopCopyToStream ROP Request Buffer

The following descriptions define valid fields for the **RopCopyToStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.8.1).

ByteCount (8 bytes): An integer that specifies the number of bytes to be copied from the source stream to the destination stream.

2.2.21.2 RopCopyToStream ROP Response Buffer

The following descriptions define valid fields for the **RopCopyToStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.8.2).

DestHandleIndex (4 bytes): An integer that specifies the location in the Server object handle table where the handle for the destination Server object is stored. The **DestHandleIndex** field MUST be set to the value of the **DestHandleIndex** field of the ROP request buffer. The

DestHandleIndex field MUST NOT be present if the **ReturnValue** field is set to any value other than NullDestinationObject (0x00000503).

ReadByteCount (8 bytes): An integer that specifies the number of bytes read from the source stream. If the **ReturnValue** field is set to NullDestinationObject (0x00000503), the value of the **ReadByteCount** field is undefined.

WrittenByteCount (8 bytes): An integer that specifies the number of bytes written to the destination stream. If the **ReturnValue** field is set to NullDestinationObject (0x00000503), the value of the **WrittenByteCount** field is undefined.

2.2.22 RopProgress ROP

The **RopProgress** ROP ([MS-OXCROPS] section 2.2.8.13) reports the progress status of an asynchronous operation. This ROP also can abort an in-progress operation if the **WantCancel** field is set to nonzero. This operation is valid on any object on which an asynchronous operation is used.

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].

2.2.22.1 RopProgress ROP Request Buffer

The following descriptions define valid fields for the **RopProgress** ROP request buffer ([MS-OXCROPS] section 2.2.8.13.1).

WantCancel (1 byte): A Boolean value that is zero if the client wants the current operation to continue. The value is nonzero to request that the server cancel the operation.

2.2.22.2 RopProgress ROP Response Buffer

The following descriptions define valid fields for the **RopProgress** ROP response buffer ([MS-OXCROPS] section 2.2.8.13.2).

CompletedTaskCount (4 bytes): An integer that specifies the approximate number of tasks that the server has completed.

TotalTaskCount (4 bytes): An integer that specifies the approximate number of tasks the server will complete for the entire operation. The value of this field MUST be greater than or equal to the value of the **CompletedTaskCount** field.

2.2.23 RopLockRegionStream ROP

The **RopLockRegionStream** ROP ([MS-OXCROPS] section 2.2.9.9) is used to lock a specified range of bytes in a Stream object. This operation is valid only on Stream objects.

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].

2.2.23.1 RopLockRegionStream ROP Request Buffer

The following descriptions define valid fields for the **RopLockRegionStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.9.1).

RegionOffset (8 bytes): An unsigned 64-bit integer that specifies the number of bytes from the beginning of the stream where the beginning of the region to be locked is located.

RegionSize (8 bytes): An unsigned 64-bit integer that specifies the size, in bytes, of the region to be locked.

LockFlags (4 bytes): A set of bits that control the read/write access to the locked region. The **LockFlags** field MUST be set to one of the values in the following table.

Value	Meaning
0x00000001	If this lock is granted, the specified range of bytes can be opened and read any number of times, but writing to the locked range is prohibited except for the owner that was granted this lock.
Any other value	If this lock is granted, reading and writing to the specified range of bytes is prohibited except by the owner that was granted this lock.

2.2.23.2 RopLockRegionStream ROP Response Buffer

This protocol adds no additional field details to the **RopLockRegionStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.9.2).

2.2.24 RopUnlockRegionStream ROP

The **RopUnlockRegionStream** ROP ([MS-OXCROPS] section 2.2.9.10) unlocks a specified range of bytes in a Stream object. This operation is valid only on Stream objects.

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].

2.2.24.1 RopUnlockRegionStream ROP Request Buffer

The following descriptions define valid fields for the **RopUnlockRegionStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.10.1).

RegionOffset (8 bytes): An unsigned 64-bit integer that specifies the number of bytes from the beginning of the stream where the beginning of the region to be unlocked is located.

RegionSize (8 bytes): An unsigned 64-bit integer that specifies the size, in bytes, of the region to be unlocked.

LockFlags (4 bytes): A set of bits that control the read/write access to the locked region. The value of the **LockFlags** field MUST be one of the values specified in section 2.2.23.1. In addition, it MUST be set to the same value that was used in the **RopLockRegionStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.9) that was used to lock the region of bytes.

2.2.24.2 RopUnlockRegionStream ROP Response Buffer

This protocol adds no additional field details to the **RopUnlockRegionStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.10.2).

2.2.25 RopWriteAndCommitStream ROP

The **RopWriteAndCommitStream** ROP ([MS-OXCROPS] section 2.2.9.11) is functionally equivalent to **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3) followed by **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4). This operation is valid only on Stream objects. This ROP MUST NOT be used on a Stream object that is opened on a property of a Folder object.

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].

2.2.25.1 RopWriteAndCommitStream ROP Request Buffer

The following descriptions define valid fields for the **RopWriteAndCommitStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.11.1).

DataSize (2 bytes): An integer that specifies the number of bytes in the **Data** field.

Data (variable): An array of bytes that constitute the data to be written and committed to the stream.

2.2.25.2 RopWriteAndCommitStream ROP Response Buffer

The following descriptions define valid fields for the **RopWriteAndCommitStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.11.2).

WrittenSize (2 bytes): An integer that specifies the number of bytes actually written and committed to the stream.

2.2.26 RopCloneStream ROP

The **RopCloneStream** ROP ([MS-OXCROPS] section 2.2.9.12) creates a new Stream object that is a clone of another Stream object. The cloned Stream object allows access to the same bytes but has a separate seek pointer. This operation is valid only on Stream objects.

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].

2.2.26.1 RopCloneStream ROP Request Buffer

This protocol adds no additional field details to the **RopCloneStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.12.1).

2.2.26.2 RopCloneStream ROP Response Buffer

This protocol adds no additional field details to the **RopCloneStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.12.2).

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 described in this document.

Changes to a property: There are two different models for persisting the new value of a property to the database. Changes to properties of a Folder object or a Logon object are saved implicitly in the database, whereas changes to properties of a Message object are not saved until the client sends a **RopSaveChangesMessage** **ROP request** ([MS-OXCROPS] section 2.2.6.3). Changes to properties of an Attachment object are not saved until the client sends a **RopSaveChangesAttachment** ROP request ([MS-OXCROPS] section 2.2.6.15) followed by a **RopSaveChangesMessage** ROP request.

Changes to a Stream object: There are two different models for persisting the new value of property that is opened as a Stream object to the database. For a Folder object, the new value of a property that is opened as a Stream object is persisted when the client sends a **RopCommitStream** ROP request ([MS-OXCROPS] section 2.2.9.4) or the client uses the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) to close the Stream object. For a Message object and an Attachment object, the persistence of the new value is the same as when the property is changed directly.

Caching property IDs: When the client uses the **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1) to map named properties to property IDs, the client can cache the property IDs for the length of its session. A property ID is not guaranteed to be the same for a new session. A property ID that is mapped from a named property is valid on any item within the Logon object.

3.1.2 Timers

None.

3.1.3 Initialization

There is no initialization specific to this protocol. Higher layers calling this protocol MUST obtain handles to the objects required by the message syntax specified in [MS-OXCROPS] section 2.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Getting Property IDs for Named Properties

All operations (setting, getting, deleting) on the properties of an object are done with a 32-bit property tag, which is built from a 16-bit property ID and a 16-bit type code. If the predefined set of property tags does not cover a semantic need for a new client feature, the client can create a custom property by defining a named property.

The client registers a property ID for a new named property or obtains a registered property ID for an existing named property by using the **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1). The client provides a property set and an identifier in either integer or string form, and the server returns the property ID, which the client uses for any operations performed on that property. The property ID of a named property has the most significant bit set (0x8000).

If the client needs to obtain the named property to which a property ID is registered, the client obtains that named property by using the **RopGetNamesFromPropertyIds** ROP ([MS-OXCROPS] section 2.2.8.2).

3.1.4.2 Reading a Property

If any of the properties to be read are named properties, the client obtains the property IDs for those properties, as specified in section 3.1.4.1.

The client then reads the properties by using either the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3) or the **RopGetPropertiesAll** ROP ([MS-OXCROPS] section 2.2.8.4). The client checks the ROP response buffer for error codes that are returned in place of the values of the properties.

3.1.4.3 Setting a Property

If any of the properties to be set are named properties, the client obtains the property IDs for those properties, as specified in section 3.1.4.1.

The client then sets the properties by using the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6). If the value of the **PropertyProblemCount** field in the ROP response buffer is nonzero, then the client checks the array contained in the **PropertyProblems** field of the ROP response buffer to verify which properties failed to be set. Clients SHOULD NOT include read-only properties in the **PropertyValues** field of the ROP request buffer.

3.1.4.4 Getting a List of an Object's Existing Properties

To get a list of properties that are set on an object, the client uses the **RopGetPropertiesList** ROP ([MS-OXCROPS] section 2.2.8.5). The client can use this list to create an equivalent duplicate of an object by copying only the properties specified in the list. For example, if the **RopGetPropertiesList** ROP is used on a Message object, the client can create a duplicate of the Message object without considering its attachments table and recipient table by copying only the listed properties.

3.1.4.5 Reading a Property as a Stream

If the server returned **NotEnoughMemory** (0x8007000E) in place of the property value in the ROP response buffer of the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3) or **RopGetPropertiesAll** ROP ([MS-OXCROPS] section 2.2.8.4), the property is too big to fit in a single ROP. In this case, the client can read the property as a stream.

If the property to be read is a named property, the client obtains the property ID for that property, as specified in section 3.1.4.1.

The client then obtains a handle to a Stream object for the property by using the **RopOpenStream** ROP ([MS-OXCROPS] section 2.2.9.1). The client sets the **OpenModeFlags** field of the ROP request buffer to **ReadOnly**. To verify that the handle was retrieved, the client checks the **ReturnValue** field of the ROP response buffer.

If it is not necessary for the client to read from the start of the stream, the client uses the **RopSeekStream** ROP ([MS-OXCROPS] section 2.2.9.7) to set the seek pointer. The client then reads data from the stream by sending one or more **RopReadStream** ROP requests ([MS-OXCROPS] section 2.2.9.2). If the **DataSize** field of the ROP response buffer is set to the maximum value that was specified in the ROP request buffer, the client can determine whether there is more data in the stream by issuing another **RopReadStream** request. The client stops reading the stream when it has read all the data it requires or when the server returns zero in the **DataSize** field.

When the client is done with the Stream object, it releases the Stream object by using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

3.1.4.6 Setting a Property with a Stream

If the property to be set is a named property, the client obtains the property ID for that property, as specified in section 3.1.4.1.

The client then obtains a handle to the Stream object on the property by using the **RopOpenStream** ROP ([MS-OXCROPS] section 2.2.9.1). The client sets the **OpenModeFlags** field of the ROP request buffer to **ReadWrite** or **Create**. To verify that the handle was retrieved, the client checks the **ReturnValue** field of the ROP response buffer.

If it is not necessary for the client to write from the start of the stream, the client sets the seek pointer by using the **RopSeekStream** ROP ([MS-OXCROPS] section 2.2.9.7). The client then writes data to the stream by using one or more **RopWriteStream** ROP requests ([MS-OXCROPS] section 2.2.9.3).

The client saves the changes to the property of a Folder object by using the **RopCommitStream** ROP ([MS-OXCROPS] section 2.2.9.4) or by closing the Stream object with the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3). The client saves the changes to the property of a Message object by using the **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3). The client saves the changes to the property of an Attachment object by using the **RopSaveChangesAttachment** ROP ([MS-OXCROPS] section 2.2.6.15) followed by the **RopSaveChangesMessage** ROP.

When the client is done with the Stream object, the client releases the Stream object by using the **RopRelease** ROP.

3.1.5 Message Processing Events and Sequencing Rules

With the exception of the **RopProgress** ROP ([MS-OXCROPS] section 2.2.8.13), all of the ROPs specified in section 2 of this specification are sent by the client and processed by the server. The client SHOULD process the ROP response buffer associated with each message it sends.

If the client set the **WantAsynchronous** field to nonzero in the ROP request buffer of any of the following ROPs, the server can respond with a **RopProgress** **ROP response**.

- **RopCopyTo** ([MS-OXCROPS] section 2.2.8.12)
- **RopCopyProperties** ([MS-OXCROPS] section 2.2.8.11)
- **RopSetReadFlags** ([MS-OXCROPS] section 2.2.6.10)
- **RopMoveCopyMessages** ([MS-OXCROPS] section 2.2.4.6)
- **RopMoveFolder** ([MS-OXCROPS] section 2.2.4.7)
- **RopCopyFolder** ([MS-OXCROPS] section 2.2.4.8)
- **RopHardDeleteMessagesAndSubfolders** ([MS-OXCROPS] section 2.2.4.10)
- **RopEmptyFolder** ([MS-OXCROPS] section 2.2.4.9)
- **RopDeleteMessages** ([MS-OXCROPS] section 2.2.4.11)
- **RopHardDeleteMessages** ([MS-OXCROPS] section 2.2.4.12)

The server can notify the client of its current progress by responding with a **RopProgress** ROP response buffer. When the client receives a **RopProgress** response, it can use the values of the **CompletedTaskCount** and **TotalTaskCount** fields to provide progress information to the user. The

client can request additional status or abort the operation by sending additional **RopProgress** ROP requests. In response to the client's **RopProgress** ROP request, the server MUST respond with either the response to the original ROP or another **RopProgress** ROP response. If the client sends a ROP other than **RopProgress** to the server with the same logon before the asynchronous operation is complete, the server MUST abort the asynchronous operation and respond to the new ROP.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

The server abstract data model is the same as the client abstract data model specified in section 3.1.1.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Processing RopGetPropertiesSpecific

When the server receives a **RopGetPropertiesSpecific** ROP request buffer ([MS-OXCROPS] section 2.2.8.3) from the client, the server parses the buffer. The server responds with a **RopGetPropertiesSpecific** 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 MUST return the values for the explicitly requested properties on the object, including those set by any client, those set by any server, and computed properties. The properties to be returned are specified in the **PropertyTags** field of the ROP request buffer. The server MUST order properties in the **PropertyValues** field of the ROP response buffer in the same order in which properties are specified in the **PropertyTags** field.

If the **WantUnicode** field is set to a nonzero value, the server MUST return string properties that are requested without a specified type (**PtypUnspecified**) in Unicode format. If the **WantUnicode** field is set to zero, the server MUST return string properties that are requested without a specified type (**PtypUnspecified**) in MBCS format. Properties requested with a specific string type MUST be returned using that type.

For properties on Message objects, the **code page** used for strings in MBCS format MUST be the code page that was set on the Message object when it was opened. If no code page was set on the Message object, the code page of the Logon object MUST be used. For properties on Attachment objects, the code page used for strings in MBCS format MUST be the code page that was set on the parent Message object when it was opened. If no code page was set on the parent Message object, the code page of the Logon object MUST be used. For all other objects, the code page used for strings in MBCS format MUST be the code page of the Logon object.

The server SHOULD<4> ignore the **PropertySizeLimit** field of the ROP request buffer.

3.2.5.2 Processing RopGetPropertiesAll

When the server receives a **RopGetPropertiesAll** ROP request buffer ([MS-OXCROPS] section 2.2.8.4) from the client, the server parses the buffer. The server responds with a **RopGetPropertiesAll** ROP response buffer. If this is the first ROP in the request buffer, and the maximum size of a ROP response buffer is used, and the serialized response would not fit, the server SHOULD<5> fail the **EcDoRpcExt2** method with a return value of 0x000003F0 or fail the **Execute** request type with a value of 0x000003F0 in the **StatusCode** field. 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 behavior for this ROP is the same as that for the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3), as specified in section 3.2.5.1, except that the server MUST return the values for all properties on the object.

3.2.5.3 Processing RopGetPropertiesList

When the server receives a **RopGetPropertiesList** ROP request buffer ([MS-OXCROPS] section 2.2.8.5) from the client, the server parses the buffer. The server responds with a **RopGetPropertiesList** 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 MUST return the list of all properties that are currently set on the object.

3.2.5.4 Processing RopSetProperties

When the server receives a **RopSetProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.6) from the client, the server parses the buffer. The server responds with a **RopSetProperties** 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 MUST modify the value of each client-writeable property specified in the **PropertyValues** field of the ROP request buffer. If the server fails to modify any client-writeable property, that property, along with details of the failure, is specified in a **PropertyProblem** structure ([MS-OXCDATA] section 2.7) that is contained in the **PropertyProblems** field of the ROP response buffer. If the **PropertyValues** field of the ROP request buffer includes any property that is read-only for the client, the server can respond to the client as follows, but the server MUST NOT set the property to the client-specified value:

- Include a **PropertyProblem** structure for the read-only property in the **PropertyProblems** field of the response.
- Fail the **RopSaveChangesMessage** ROP request ([MS-OXCROPS] section 2.2.6.3) when the client attempts to commit the property setting on a Message object.

- Fail either the **RopSaveChangesMessage** ROP request or the **RopSaveChangesAttachment** ROP request ([MS-OXCROPS] section 2.2.6.15) when the client attempts to commit the property setting on an Attachment object.
- Disregard the read-only property. In this case, the server gives no indication that the client's request to modify the read-only property failed.

For Message objects, the new value of the properties MUST be made available immediately for retrieval by a ROP that uses the same Message object handle. For example, if the client uses the same object handle in a **RopGetPropertiesAll** ROP request ([MS-OXCROPS] section 2.2.8.4) to read those same properties, the modified value MUST be returned. However, the modified value MUST NOT be persisted to the database until a successful **RopSaveChangesMessage** ROP is issued.

For Attachment objects, the new value of the properties MUST be made available immediately for retrieval by a ROP that uses the same Attachment object handle. However, the modified value MUST NOT be persisted to the database until a successful **RopSaveChangesAttachment** ROP followed by a successful **RopSaveChangesMessage** ROP is issued.

For Folder objects and Logon objects, the new value of the properties MUST be persisted immediately without requiring another ROP to commit it.

3.2.5.5 Processing RopDeleteProperties

When the server receives a **RopDeleteProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.8) from the client, the server parses the buffer. The server responds with a **RopDeleteProperties** 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 MUST delete the property from the object. If the server fails to delete a property, that property, along with details of the failure, is specified in a **PropertyProblem** structure ([MS-OXCDATA] section 2.7) that is contained in the **PropertyProblems** field of the ROP response buffer.

If the server returns success, it MUST NOT have a valid value to return to a client that asks for the value of this property. If a client requests the value of this property, the server SHOULD return the NotFound error (0x8004010F) ([MS-OXCDATA] section 2.4.2) in place of a value.

For Message objects, the properties MUST be removed immediately when using the same handle. In other words, if the client uses the same handle to read those same properties using the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3) or the **RopGetPropertiesAll** ROP ([MS-OXCROPS] section 2.2.8.4), the properties MUST be deleted. However, the deleted properties MUST NOT be persisted to the database until a successful **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) is issued.

For Attachment objects, the properties MUST be removed immediately when using the same handle. However, the deleted properties MUST NOT be persisted to the database until a successful **RopSaveChangesAttachment** ROP ([MS-OXCROPS] section 2.2.6.15) and a successful **RopSaveChangesMessage** ROP are issued, in that order.

For Folder objects and Logon objects, the properties MUST be removed immediately without requiring another ROP to commit the change.

3.2.5.6 Processing RopQueryNamedProperties

When the server receives a **RopQueryNamedProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.10) from the client, the server parses the buffer. The server responds with a **RopQueryNamedProperties** 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 MUST return the list of all named properties and their property IDs, filtered based on the fields in the ROP request buffer. Starting with the full list of all named properties:

1. If the **NoStrings** bit is set in the **QueryFlags** field of the ROP request buffer, named properties with the **Kind** field ([MS-OXCDATA] section 2.6.1) set to 0x1 MUST NOT be returned.

2. If the **NoIds** bit is set in the **QueryFlags** field, named properties with the **Kind** field set to 0x0 MUST NOT be returned.

3. If the **PropertyGuid** field of the ROP request buffer is present, named properties with a **GUID** field ([MS-OXCDATA] section 2.6.1) value that does not match the value of the **PropertyGuid** field MUST NOT be returned.

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

3.2.5.7 Processing RopCopyProperties

When the server receives a **RopCopyProperties** ROP request buffer ([MS-OXCROPS] section 2.2.8.11) from the client, the server parses the buffer. The server responds with a **RopCopyProperties** 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 MUST copy or move the properties specified from the source object to the destination object. If the server fails to copy or move a property, that property, along with details of the failure, is specified in a **PropertyProblem** structure ([MS-OXCDATA] section 2.7) that is contained in the **PropertyProblems** field of the ROP response buffer.

If the **Move** flag is set in the **CopyFlags** field of the ROP request buffer, the server MAY<6> delete the copied properties from the source object. If the **NoOverwrite** flag is set in the **CopyFlags** field, the server MUST NOT overwrite any properties that already have a value on the destination object. If any other bits are set in the **CopyFlags** field, the server SHOULD<7> return an InvalidParameter error (0x80070057) ([MS-OXCDATA] section 2.4).

In the case of Message objects, the changes on either source or destination MUST NOT be persisted until the **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) is successfully issued. In the case of Attachment objects, the changes on either source or destination MUST NOT be persisted until the **RopSaveChangesAttachment** ROP ([MS-OXCROPS] section 2.2.6.15) and the **RopSaveChangesMessage** ROP are successfully issued, in that order.

In the case of Folder objects, the changes on the source and destination MUST be immediately persisted. If the original object is a Folder object and the **CopyFlags** field has the **Move** flag set, the server SHOULD return a NotSupported error (0x80040102) ([MS-OXCDATA] section 2.4).

If the client requests asynchronous processing, the server can process this ROP asynchronously. During asynchronous processing, the server can indicate that the operation is still being processed by returning a **RopProgress** ROP response ([MS-OXCROPS] section 2.2.8.13), or it can indicate that the operation has already completed by returning a **RopCopyProperties** ROP response. If the operation fails at any point during the asynchronous processing, the server returns a **RopCopyProperties** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see section 2.2.22 and section 3.2.5.19.

3.2.5.8 Processing RopCopyTo

When the server receives a **RopCopyTo** ROP request buffer ([MS-OXCROPS] section 2.2.8.12) from the client, the server parses the buffer. The server responds with a **RopCopyTo** 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 MUST copy or move only the writable properties that are not specified in the **ExcludedTags** field.

If the **Move** flag is set in the **CopyFlags** field of the ROP request buffer, the server MAY<8> delete the copied properties from the source object. If the **NoOverwrite** flag is set in the **CopyFlags** field, the server MUST NOT overwrite any properties that already have a value on the destination object. If any other bits are set in the **CopyFlags** field, the server SHOULD<9> return an InvalidParameter error (0x80070057).

In the case of Message objects, the changes on either source or destination MUST NOT be persisted until the **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) is successfully issued. In the case of Attachment objects, the changes on either source or destination MUST NOT be persisted until the **RopSaveChangesAttachment** ROP ([MS-OXCROPS] section 2.2.6.15) and the **RopSaveChangesMessage** ROP are successfully issued, in that order.

In the case of Folder objects, the changes on the source and destination MUST be immediately persisted. If the original object is a Folder object and the **CopyFlags** field has the **Move** flag set, the server MAY return a NotSupported error (0x80040102). The server SHOULD<10> return a NotSupported error if the **RopCopyTo** ROP request specifies a **public folder** as either the source object or the destination object.

If the client requests asynchronous processing, the server can process this ROP asynchronously. During asynchronous processing, the server can indicate that the operation is still being processed by returning a **RopProgress** ROP response ([MS-OXCROPS] section 2.2.8.13), or it can indicate that the operation has already completed by returning a **RopCopyTo** ROP response. If the operation fails at any point during the asynchronous processing, the server returns a **RopCopyTo** ROP response buffer with an appropriate error code. For details about the **RopProgress** ROP and how it is used, see section 2.2.22 and section 3.2.5.19.

The following error codes SHOULD be returned when the scenarios described in the corresponding Description column of the following table are met.

Error code name	Value	Description
NotSupported	0x80040102	The source object and destination object are not compatible with each other for the copy operation. The source object and destination object need to be of the same type, and MUST be a Message object, Folder object, or Attachment object.
MessageCycle	0x00000504	The source message directly or indirectly contains the destination message.
FolderCycle	0x8004060B	The source folder contains the destination folder.
CollidingNames	0x80040604	A subobject cannot be copied because there is already a subobject existing in the destination object with the same display name, which is specified in the **PidTagDisplayName** property ([MS-OXCFOLD] section 2.2.2.2.2.5), as the subobject to be copied.

3.2.5.9 Processing RopGetNamesFromPropertyIds

When the server receives a **RopGetNamesFromPropertyIds** ROP request buffer ([MS-OXCROPS] section 2.2.8.2) from the client, the server parses the buffer. The server responds with a **RopGetNamesFromPropertyIds** 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.

For each property ID in the **PropertyIds** field of the ROP request buffer, the server MUST perform the following:

1. If the property ID is less than 0x8000, the associated **PropertyName** structure ([MS-OXCDATA] section 2.6.1) contained in the **PropertyNames** field of the ROP response buffer MUST be composed as follows:
   - The structure's **GUID** field is set to the PS_MAPI property set ([MS-OXPROPS] section 1.3.2).
   - The structure's **Kind** field is set to 0x00.
   - The structure's **LID** field is set to the property ID.
2. For property IDs that have an associated **PropertyName** structure, the server MUST return the **PropertyName** structure associated with theproperty ID.
3. For property IDs that do not have an associated **PropertyName** structure, the associated name MUST be composed as follows:
   - A value in the **Kind** field of 0xFF. There is no other return data for this entry.

3.2.5.10 Processing RopGetPropertyIdsFromNames

When the server receives a **RopGetPropertyIdsFromNames** ROP request buffer ([MS-OXCROPS] section 2.2.8.1) from the client, the server parses the buffer. The server responds with a **RopGetPropertyIdsFromNames** 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 **PropertyNameCount** field of the ROP request buffer is zero, and the **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1) is acting on a Logon object, the server MUST enumerate all property IDs that are registered for named properties. Otherwise, the server MUST, for each entry in the **PropertyNames** field of the ROP request buffer, follow this procedure:

1. If the **GUID** field of the **PropertyName** structure ([MS-OXCDATA] section 2.6.1) in the ROP request buffer specifies the PS_MAPI property set, the returned property ID is obtained from the **LID** field. Otherwise, if the **GUID** field specifies the PS_INTERNET_HEADERS property set and the **Kind** field of the **PropertyName** structure is set to 0x01, coerce the value of the **Name** field to all lowercase. Property sets are specified in [MS-OXPROPS] section 1.3.2.
2. Find the property ID registered for the named property that matches the **PropertyName** structure as follows:
   - If the **Kind** field is set to 0x01, the named property has a property name that matches the value of the **Name** field of the **PropertyName** structure.
   - If the **Kind** field is set to 0x00, the named property has a LID that matches the value of the **LID** field of the **PropertyName** structure.
3. For unfound rows, the returned property ID MUST be 0x0000 unless all of the following conditions are true:
   - The **Flags** field of the ROP request buffer is set to 0x02.
   - The user has permission to create new entries.

- The server-imposed limit on property ID mappings specified later in this section hasn't yet been reached.

4. If the above conditions in step three are all met, a new property ID is registered for the named property. The newly assigned property ID MUST be unique in that it MUST NOT be assigned to another named property and MUST NOT be equal to 0xFFFF. The newly assigned property ID MUST be greater than 0x8000.

Because only 32,767 property IDs are available (15 significant bits), the server MUST impose a limit of at most 32,767 on the total number of registered property IDs. If the client attempts to register additional property IDs and the server has reached this limit, then the server MUST return **OutOfMemory** (0x8007000E) in the **ReturnValue** field of the ROP response buffer.

3.2.5.11 Processing RopOpenStream

When the server receives a **RopOpenStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.1) from the client, the server parses the buffer. The server responds with a **RopOpenStream** 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 MUST open the stream in the mode indicated by the **OpenModeFlags** field as specified by the table insection 2.2.14.1. The server SHOULD<11> return a NotSupported error (0x80040102) if the client attempts to open a stream on a public folder in any mode other than ReadOnly.

The implementation of the **RopOpenStream** ROP can allocate some temporary resource on the server to represent the link between the Folder object returned to the client and the Folder object in the database. To free this resource, the client MUST call the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) on the Folder object.

The server MUST store the location of the seek pointer until the client calls the **RopRelease** ROP. The seek pointer MUST be initialized as specified in the request. The stream MUST be prepopulated with the current value of the property that is specified in the request.

The following error code SHOULD be returned when the scenario described in the Description column of the following table is met.

Error code name	Value	Description
NotSupported	0x80040102	A stream cannot be opened on the specified object.
NotFound	0x8004010F	The property tag does not exist for the object, and it cannot be created because the **Create** bit was not specified in **OpenModeFlags** field.

3.2.5.12 Processing RopReadStream

When the server receives a **RopReadStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.2) from the client, the server parses the buffer. The server responds with a **RopReadStream** 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 MUST read less than or equal to the amount of data requested. If the **ByteCount** field of the ROP request buffer is set to 0xBABE, the number of bytes read MUST be less than or equal to the

value of the **MaximumByteCount** field of the ROP request buffer; otherwise, the number of bytes read MUST be less than or equal to the value of the **ByteCount** field.

The server MUST read from the Stream object beginning at the current seek pointer. The seek pointer MUST be moved forward the same number of bytes as was read from the Stream object. In the case of a failure, the **DataSize** field SHOULD be set to zero.

3.2.5.13 Processing RopWriteStream

When the server receives a **RopWriteStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.3) from the client, the server parses the buffer. The server responds with a **RopWriteStream** 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 SHOULD<12> return the TooBig error code if it writes less than the amount requested. The server MUST store the data that is specified in the ROP request buffer into a stream buffer, writing forward starting at the current seek pointer. The seek pointer MUST be moved forward the same number of bytes as were written to the Stream object.

After a **RopWriteStream** ROP request buffer is processed, the new value of the property MUST be immediately available for retrieval by a ROP that uses the same object handle. However, the new value of the property is not persisted to the database. For a Message object, the new value is persisted to the database when a successful **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) is issued. For an Attachment object, the new value is persisted to the database when a successful **RopSaveChangesAttachment** ROP ([MS-OXCROPS] section 2.2.6.15) followed by a successful **RopSaveChangesMessage** ROP is issued. For a Folder object, the value is persisted when the **RopCommitStream** ROP ([MS-OXCROPS] section 2.2.9.4) is issued on the Stream object or the Stream object is closed with a **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

The following error codes SHOULD be returned when the scenarios described in the corresponding Description column of the following table are met.

Error code name	Value	Description
TooBig	0x80040305	The write will exceed the maximum stream size.
StreamAccessDenied	0x80030005	Write access is denied.

3.2.5.14 Processing RopCommitStream

When the server receives a **RopCommitStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.4) from the client, the server parses the buffer. The server responds with a **RopCommitStream** 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 MUST set the property specified in the **RopOpenStream** ROP request ([MS-OXCROPS] section 2.2.9.1) with the data from the Stream object.

3.2.5.15 Processing RopGetStreamSize

When the server receives a **RopGetStreamSize** ROP request buffer ([MS-OXCROPS] section 2.2.9.5) from the client, the server parses the buffer. The server responds with a **RopGetStreamSize** 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 MUST return the current size of the Stream object.

3.2.5.16 Processing RopSetStreamSize

When the server receives a **RopSetStreamSize** ROP request buffer ([MS-OXCROPS] section 2.2.9.6) from the client, the server parses the buffer. The server responds with a **RopSetStreamSize** 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 sets the current size of the Stream object according to the value specified in the **StreamSize** field of the ROP request buffer. If the size of the stream is increased, the server MUST set the values in the extended stream to 0x00. If the size of the stream is decreased, the server discards the values that are beyond the end of the new size.

3.2.5.17 Processing RopSeekStream

When the server receives a **RopSeekStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.7) from the client, the server parses the buffer. The server responds with a **RopSeekStream** 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 modifies the location of the seek point associated with the Stream object according to the ROP request buffer. If the client requests the seek pointer be moved beyond 2^31 bytes, the server MUST return the StreamSeekError error code in the **ReturnValue** field of the ROP response buffer. If the client requests the seek pointer be moved beyond the end of the stream, the stream is extended, and zeros filled to the new seek location.

The following error codes SHOULD be returned when the scenarios described in the corresponding Description column of the following table are met.

Error code name	Value	Description
StreamSeekError	0x80030019	Tried to seek to offset before the start or beyond the max stream size of 2^31.
StreamInvalidParam	0x80030057	The value of the **Origin** field is invalid.

3.2.5.18 Processing RopCopyToStream

When the server receives a **RopCopyToStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.8) from the client, the server parses the buffer. The server responds with a **RopCopyToStream** 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.

Servers SHOULD<13> implement this ROP as follows: The server MUST read the number of bytes requested from the source Stream object and write those bytes into the destination Stream object. The server MUST move the seek pointer of both the source and destination streams forward the same number of bytes as were copied.

The following error code SHOULD be returned when the scenario described in the Description column of the table is met.

Error code name	Value	Description
NullDestinationObject	0x00000503	Destination object does not exist.

3.2.5.19 Processing RopProgress

When the server receives a **RopProgress** ROP request buffer ([MS-OXCROPS] section 2.2.8.13) from the client, the server parses the buffer. The server responds with a **RopProgress** 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.

Servers SHOULD<14> implement this ROP as follows:

If the server implements this ROP, the server MUST respond with a **RopProgress** ROP response when the server is not done with the asynchronous operation. If the server receives a **RopProgress** ROP request with the **WantCancel** field set to nonzero, then the server can abort the operation instead of completing it. If the server has completed or aborted the operation, it MUST respond with a ROP response buffer that corresponds to the original ROP request.

3.2.5.20 Processing RopLockRegionStream

When the server receives a **RopLockRegionStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.9) from the client, the server parses the buffer. The server responds with a **RopLockRegionStream** 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.

Servers MAY<15> implement this ROP as follows:

If the server implements this ROP, the server MUST check for any existing locks on the requested region of the Stream object. If any are found, the server MUST check the last activity time on the session that locked the region previously. If the last activity time was greater than 30 seconds prior to the new request, the server MUST mark the previous lock as expired and discard any pending changes of the Stream object from the session that owned that lock. If all previous locks are expired, or if there are no previous locks, the server MUST grant the requesting client a new lock. If there are previous locks that are not expired, the server MUST return the AccessDenied error code (0x80070005) in the **ReturnValue** field of the ROP response buffer.

If a session with an expired lock calls any ROP for the Stream object that would encounter the locked region, the server MUST return the NetworkError error code (0x80040115) in the **ReturnValue** field of the ROP response buffer. The server MUST limit access to the locked region by other sessions as indicated by the **LockFlags** field of the ROP request buffer.

3.2.5.21 Processing RopUnlockRegionStream

When the server receives a **RopUnlockRegionStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.10) from the client, the server parses the buffer. The server responds with a **RopUnlockRegionStream** 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.

Servers MAY<16> implement this ROP as follows:

If the server implements this ROP, the server MUST remove any existing locks on the requested region that are owned by the session calling the ROP. If there are previous locks that are not owned by the session calling the ROP, the server MUST leave them unmodified.

3.2.5.22 Processing RopWriteAndCommitStream

When the server receives a **RopWriteAndCommitStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.11) from the client, the server parses the buffer. The server responds with a **RopWriteAndCommitStream** 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.

Servers MAY<17> implement this ROP. If the server implements this ROP, it MUST first process the ROP request as specified in section 3.2.5.13 and then process the ROP request as specified in section 3.2.5.14.

3.2.5.23 Processing RopCloneStream

When the server receives a **RopCloneStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.12) from the client, the server parses the buffer. The server responds with a **RopCloneStream** 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.

Servers MAY<18> implement this ROP as follows:

If the server implements this ROP, it MUST create a new Stream object that contains the same data as the source Stream object. Changes made to the stream in one Stream object MUST be immediately visible in the other. The initial setting of the seek pointer in the cloned Stream object MUST be the same as the current setting of the seek pointer in the source Stream object at the time that this ROP is processed. Afterwards, the seek pointers MUST be independent of each other.

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. Please note that the examples listed here only show 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 for a multibyte field appear in **little-endian** format, with the bytes in the field presented from least significant to most significant. Generally speaking, these ROP requests are packed with other ROP requests, compressed and obfuscated, as described in [MS-OXCRPC] section 3. These examples assume the client has already successfully logged on to the server and 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.

```
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.

Such a byte sequence is then followed by one or more lines interpreting 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.

The following example shows how a property tag and its value are represented in a buffer and interpreted directly from it (according to the property buffer format described in [MS-OXCDATA] section 2.2.3). The property tag appears in the buffer in little-endian format.

```
0021: 03 00 76 66 0A 00 00-00
```

PropertyTag: 0x66760003 (**PidTagRuleSequence** ([MS-OXORULE] section 2.2.1.3.1.2))

PropertyValue: 10

Generally speaking interpreted values will be shown in their native format, interpreted appropriately from the raw byte sequence as described in the appropriate section. Here, the byte sequence 0A 00 00 00 has been interpreted as a **PtypInteger32** ([MS-OXCDATA] section 2.11.1) with a value of 10 because the type of the **PidTagRuleSequence** property is **PtypInteger32**.

4.1 Getting Property IDs

The getting PropertyIds example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopGetPropertyIdsFromNames** ROP as described in section 2.2.12.

4.1.1 Client Request Buffer

The client requests the server to register property IDs for two named properties it will create on a Message object.

A complete ROP request buffer is formatted as follows.

```
0000: 56 00 00 02 02 00 01 02-20 06 00 00 00 00 00 c0
```

```
0010: 00 00 00 00 00 00 46 14-54 00 65 00 73 00 74 00
0020: 50 00 72 00 6f 00 70 00-31 00 00 00 01 02 20 06
0030: 00 00 00 00 00 c0 00 00-00 00 00 00 46 14 54 00
0040: 65 00 73 00 74 00 50 00-72 00 6f 00 70 00 32 00
0050: 00 00
```

The first three bytes of the buffer refer to the **RopId, LogonId**, and **InputHandleIndex** fields of the **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1).

```
0000: 56 00 00
```

RopId: 0x56 (**RopGetPropertyIdsFromNames**)

LogonId: 0x00

InputHandleIndex: 0x00

The next three bytes refer to the **Flags** and **PropertyNameCount** fields defined in section 2.2.12.1.

```
0003: 02 02 00
```

Flags: 0x02. **Create** flag, indicating that server will create new property IDs for any named properties that do not already have an existing mapping.

PropertyNameCount: 0x0002. Two properties in the **PropertyNames** field follow.

The remaining bytes form the **PropertyNames** field. Each row in the **PropertyNames** field contains a **PropertyName** structure ([MS-OXCDATA] section 2.6).

```
0006: 01 02-20 06 00 00 00 00 00 C0
0010: 00 00 00 00 00 00 46 14-54 00 65 00 73 00 74 00
0020: 50 00 72 00 6F 00 70 00-31 00 00 00 01 02 20 06
0030: 00 00 00 00 00 C0 00 00-00 00 00 00 46 14 54 00
0040: 65 00 73 00 74 00 50 00-72 00 6F 00 70 00 32 00
0050: 00 00
```

Property 1:

Kind: 0x01 MNID_STRING

GUID: 00062002-0000-0000-C000-000000000046

NameSize: 0x14

Name: TestProp1

Property 2:

Kind: 0x01 MNID_STRING

GUID: 00062002-0000-0000-C000-000000000046

NameSize: 0x14

Name: TestProp2

4.1.2 Server Response Buffer

The server registers two named properties on the Message object and sends a response to the client.

A complete ROP response buffer is formatted as follows.

```
0000: 56 00 00 00 00 00 02 00-3E 86 3F 86
```

The first six bytes of the response buffer contain the **RopId**, **InputHandleIndex**, and **ReturnValue** fields, as described in [MS-OXCROPS] section 2.2.

```
0000: 56 00 00 00 00 00
```

RopId: 0x56 (**RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1))

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

The next two bytes specify the number of property IDs that are packed in the buffer as described in section 2.2.12.2.

PropertyIdCount: 0x0002 (2 properties in the **PropertyIds** field)

The remaining bytes are for the **PropertyIds** field, where each entry in the array is four-byte property ID as described in section 2.2.12.2.

```
0009: 3E 86 3F 86
```

property ID: 0x863E

property ID: 0x863F

4.2 Setting Properties

The setting properties example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) as described in section 2.2.5.

4.2.1 Client Request Buffer

The client requests the server to set values for two properties on a Message object.

A complete ROP request buffer is a variable length sequence, with seven required bytes followed by a property value array. An example of the request buffer follows.

```
0000: 0A 00 00 24 00 02 00 1F-00 3D 00 00 00 1F 00 1D
0010: 0E 48 00 65 00 6C 00 6C-00 6F 00 20 00 57 00 6F
0020: 00 72 00 6C 00 64 00 00-00
```

The first three bytes of the buffer refer to the **RopId, LogonId**, and **InputHandleIndex** fields of the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6).

```
0000: 0A 00 00
```

RopId: 0x0A (**RopSetProperties**)

LogonId: 0x00

InputHandleIndex: 0x00

The next four bytes refer to the **PropertyValueSize** and **PropertyValueCount** fields of the **RopSetProperties** ROP defined in section 2.2.5.1. For more details on property buffer format, see [MS-OXCDATA] section 2.2.3.

```
0003:          24 00 02 00
```

PropertyValueSize: 0x0024. There are 36 (decimal) bytes contained in both the **PropertyValueCount** field and the **PropertyValues** field.

PropertyValueCount: 0x0002. Two properties in the property array.

The remaining bytes constitute the **PropertyValues** field, where each row in the array consists of property tag and value. A property tag is a 32-bit number that contains a property type in bits 0 through 15 and a unique property ID in bits 16 through 31 as shown in the following illustration.

```
0007:                      1F-00 3D 00 00 00 1F 00 1D
0010: 0E 48 00 65 00 6C 00 6C-00 6F 00 20 00 57 00 6F
0020: 00 72 00 6C 00 64 00 00-00
```

Property 1 (0008: 1F 00 3D 00 00 00)

PropertyTag: 0x003D001F (**PropertyId**: 0x003D->**PidTagSubjectPrefix**, **PropertyType**: 0x001F-> **PtypString**)

PropertyValue: 0x0000 (empty string)

Property 2 (000D: 1F 00 1D 0E 48 00 65 00 6C 00 6C 00 6F 00 20 00 57 00 6F 00 72 00 6C 00 64 00 00 00)

PropertyTag: 0x0E1D001F (**PropertyId**: 0x0E1D->**PidTagNormalizedSubject**, **PropertyType**: 0x001F-> PtypString)

PropertyValue: 48 00 65 00 6C 00 6C 00 6F-00 20 00 57 00 6F 00 72 00 6C 00 64 00 00 00 (Hello World)

4.2.2 Server Response Buffer

The server sets the values for the two properties on the Message object and sends a response to the client.

A complete ROP response buffer is formatted as follows.

```
0000: 0A 00 00 00 00 00 00 00
```

The first six bytes of the **RopSetProperties** ROP response buffer ([MS-OXCROPS] section 2.2.8.6) contain the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

RopId: 0x0A (**RopSetProperties**)

InputHandleIndex: 0x00

ReturnValue: 0x00000000. (ecNone: Success)

The final two bytes in the ROP response buffer are the **PropertyProblemCount** field described in section 2.2.5.2.

```
0006: 00 00
```

PropertyProblemCount: 0x0000. Count of problem properties that follows. A value of 0 indicates that all properties were set successfully.

4.3 Getting Properties

The getting properties example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3) as described in section 2.2.2.

4.3.1 Client Request Buffer

The client is requesting property values for specific properties from the server. The first two are named properties and the third one is a standard property. The property IDs for the named properties were gotten from the server by calling the **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1). For more details, see section 4.1 .

A complete **RopGetPropertiesSpecific** ROP request buffer ([MS-OXCROPS] section 2.2.8.3) is a variable length sequence, with nine required bytes followed by a property tag array. The following example shows the request buffer.

```
0000: 07 00 00 00 00 01 00 03-00 0B 00 3E 86 03 00 3F
0010: 86 02 01 E2 65
```

The first three bytes refer to the **RopId**, **LogonId**, and **InputHandleIndex** fields as described in [MS-OXCROPS] section 2.2.

```
0000: 07 00 00
```

RopId: 0x07 (**RopGetPropertiesSpecific** ([MS-OXCROPS] section 2.2.8.3))

LogonId: 0x00

InputHandleIndex: 0x00

The next six bytes of the request buffer are the **PropertySizeLimit**, **WantUnicode**, and **PropertyTagCount** fields as described in section 2.2.2.1.

```
0003: 00 00 01 00 03-00
```

PropertySizeLimit: 0x0000. No property size set; maximum limit is default.

WantUnicode: 0x0001. Nonzero means Unicode.

PropertyTagCount: 0x0003. Three property tags in the array.

The remaining bytes constitute the **PropertyTags** field as described in section 2.2.2.1.

```
0009: 0B 00 3E 86 03 00 3F 86 02 01 E2 65
```

PropertyTag1: 0x863E000B (**PropertyId**: 0x863E->TestProp1, **PropertyType**: 0x000B-> **PtypBoolean**)

PropertyTag2: 0x863F0003 (**PropertyId**: 0x863F->TestProp2, **PropertyType**: 0x0003-> **PtypInteger32**)

PropertyTag3: 0x65E20102 (**PropertyId**: 0x65E2->**PidTagChangeKey**, **PropertyType**: 0x0102-> **PtypBinary**)

4.3.2 Server Response Buffer

The server returns the property values to the client.

A complete ROP response buffer is formatted as follows.

```
0000: 07 00 00 00 00 00 01 00-00 00 62 00 00 00 0a 0f
0010: 01 04 80
```

The first six bytes of the **RopGetPropertiesSpecific** ROP response buffer ([MS-OXCROPS] section 2.2.8.3) contain the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

```
0000: 07 00 00 00 00 00
```

RopId: 0x07 (**RopGetPropertiesSpecific** ([MS-OXCROPS] section 2.2.8.3))

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

The remaining bytes in the ROP response buffer are for the **RowData** field, which contains a **PropertyRow** structure. The first byte of the **RowData** field is the **Flag** field of the **PropertyRow** structure. This is followed by three **FlaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.5). The order of properties is the same as in the request buffer.

```
0006: 01 00 00 00 62 00 00 00-0A 0F 01 04 80
```

Flag: 0x01 (Nonzero indicates there was an error in at least one of the property values)

Property 1:

Flag: 0x00

PropertyValue: 0x00 (False)

Property 2:

Flag: 0x00

PropertyValue: 0x00000062 (98 in decimal)

Property 3:

Flag: 0x0A (Indicates that the **PropertyValue** field will be an error code)

PropertyValue: 0x8004010F (NotFound)

4.4 Working with Streams

Examples of how to set a value for a property using streams are given in section 4.4.1 through section 4.4.3.2. These examples cover the following ROPs:

- **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1)
- **RopSetStreamSize** ([MS-OXCROPS] section 2.2.9.6)
- **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3)
- **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4)

4.4.1 Opening a Stream

The opening a stream example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopOpenStream** ROP ([MS-OXCROPS] section 2.2.9.1) as described in section 2.2.14.

4.4.1.1 Client Request Buffer

A complete **RopOpenStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.1) is nine bytes in length, formatted as follows.

```
0000: 2B 01 00 01 02 01 9A 0E-01
```

The first four bytes of the buffer refer to the **RopId**, **LogonId**, **InputHandleIndex**, and **OutputHandleIndex** fields of the **RopOpenStream** ROP.

```
0000: 2B 01 00 01
```

RopId: 0x2B (**RopOpenStream**)

LogonId: 0x01

InputHandleIndex: 0x00

OutputHandleIndex: 0x01. Index in the Server object handle table for the object created by this ROP.

The next five bytes refer to the **PropertyTag** and **OpenModeFlags** fields defined in section 2.2.14.1.

```
0004: 02 01 9A 0E-01
```

PropertyTag: 0x0E9A0102 (**PidTagExtendedRuleMessageCondition** ([MS-OXORULE] section 2.2.4.1.10))

OpenModeFlags: 0x01. ReadWrite mode

4.4.1.2 Server Response Buffer

The complete ROP response buffer is formatted as follows.

```
0000: 2B 01 00 00 00 00 15 2E-00 00
```

The first six bytes of the **RopOpenStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.1) contain the **RopId**, **OutputHandleIndex**, and **ReturnValue** fields.

```
0000: 2B 01 00 00 00 00
```

RopId: 0x2B (**RopOpenStream**)

OutputHandleIndex: 0x01

ReturnValue: 0x00000000 (Success)

The last four bytes in the ROP response buffer are the **StreamSize** field as described in section 2.2.14.2.

```
0006: 15 2E-00 00
```

StreamSize: 0x00002E15 (11797)

4.4.2 Writing to the Stream

The writing to the stream example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopWriteStream** ROP ([MS-OXCROPS] section 2.2.9.3) as described in section 2.2.16.

4.4.2.1 Client Request Buffer

A complete **RopWriteStream** ROP request buffer ([MS-OXCROPS] section 2.2.9.3) is a variable length buffer, formatted as follows.

```
0000: 2D 01 01 15 2E 00 00 61-6E 20 61 6C 77 61 79 73
0010: 20 72 65 73 74 6f 72 65-20 74 68 65 20 6C 6F 6F
0020: 6B 20 6F 66 20 79 6F 75-72 20 64 6F 63 75 6D 65
```

The first three bytes of the buffer refer to the **RopId**, **LogonId**, and **InputHandleIndex** fields of the **RopWriteStream** ROP.

```
0000: 2D 01 01
```

RopId: 0x2D (**RopWriteStream**)

LogonId: 0x01

InputHandleIndex: 0x01

The next two bytes in the ROP request buffer are the **DataSize** field, described in section 2.2.16.1.

```
0006: 15 2E
```

DataSize: 0x2E15 (11797)

The remaining bytes constitute the **Data** field. The ROP request buffer specified earlier in this section is truncated, and all of the stream data is not shown.

Data: 00 00 61-6E 20 61 6C 77 61 79 73 20 72 65 73 74 6F 72 65-20 74 68 65 20 6C 6F 6F 6B 20 6F 66 20 79 6F 75-72 20 64 6F 63 75 6D 65.....

4.4.2.2 Server Response Buffer

The complete ROP response buffer is formatted as follows.

```
0000: 2D 01 00 00 00 00 15 2E
```

The first six bytes of the **RopWriteStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.3) contain the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

```
0000: 2D 01 00 00 00 00
```

RopId: 0x2D (**RopWriteStream**)

InputHandleIndex: 0x01

ReturnValue: 0x00000000 (Success)

The last two bytes are the **WrittenSize** field described in section 2.2.16.2.

```
0006: 15 2E
```

WrittenSize: 0x2E15 (11797)

4.4.3 Committing a Stream

The committing a stream example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopCommitStream** ROP ([MS-OXCROPS] section 2.2.9.4) as described in section 2.2.17.

4.4.3.1 Client Request Buffer

For the purposes of this example, it is assumed that a stream is open before this ROP is called. A complete ROP request buffer for the **RopCommitStream** ROP ([MS-OXCROPS] section 2.2.9.4) is a three-byte sequence formatted as follows.

```
0000: 5D 01 01
```

The three bytes refer to the **RopId**, **LogonId**, and **InputHandleIndex** fields described in [MS-OXCROPS] section 2.2.

RopId: 0x5D (**RopCommitStream**)

LogonId: 0x01

InputHandleIndex: 0x01

4.4.3.2 Server Response Buffer

The complete ROP response buffer is formatted as follows.

```
0000: 5d 01 00 00 00 00
```

The six bytes of the **RopCommitStream** ROP response buffer ([MS-OXCROPS] section 2.2.9.4) contain the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

RopId: 0x5D (**RopCommitStream**)

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

4.5 Asynchronous Progress

The asynchronous progress example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopProgress** ROP ([MS-OXCROPS] section 2.2.8.13) as described in section 2.2.22. In this example, the user is trying to empty a Folder object, which has 729 items in it, and the **RopEmptyFolder** ROP ([MS-OXCROPS] section 2.2.4.9) will represent the asynchronous operation that the **RopProgress** ROP reports progress for.

The sequence in which ROPs get passed between client and server is shown in the following figure.

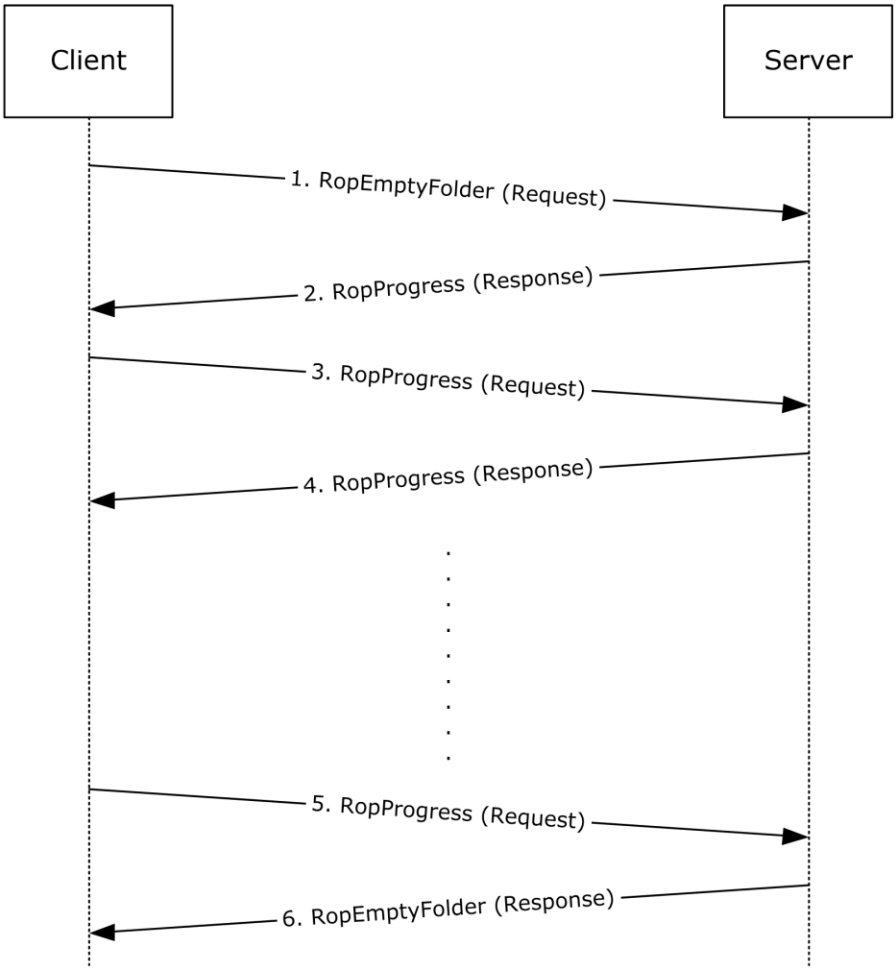

Figure 1: Sequence in which ROPs are passed between client and server

1. Client sends a **RopEmptyFolder** ROP request to the server. The ROP request buffer is formatted as follows.

0000: 58 00 00 01 00

RopId: 0x58 (**RopEmptyFolder**)

LogonId: 0x00

InputHandleIndex: 0x00

WantAsynchronous: 0x01 (TRUE)

WantDeleteAssociated: 0x00 (FALSE)

2. Server responds to request by sending the **RopProgress** ROP response buffer, which is formatted as follows.

0000: 50 00 00 00 00 00 00 1D-00 00 00 D9 02 00 00

RopId: 0x50 (**RopProgress**)

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

LogonId: 0x00

CompletedTaskCount: 0x0000001D (29 in decimal)

TotalTaskCount: 0x000002D9 (729 in decimal)

3. Now client sends a **RopProgress** ROP request buffer asking server how much progress has been made. The ROP request buffer is formatted as follows.

0000: 50 00 00 00

RopId: 0x50 (**RopProgress**)

LogonId: 0x00

InputHandleIndex: 0x00

WantCancel: 0x00 (FALSE)

4. Server responds to request by sending the **RopProgress** ROP response buffer, which is formatted as follows.

0000: 50 00 00 00 00 00 00 3B-00 00 00 D9 02 00 00

RopId: 0x50 (**RopProgress**)

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (ecNone: Success)

LogonId: 0x00

CompletedTaskCount: 0x0000003B (59 in decimal)

TotalTaskCount: 0x000002D9 (729 in decimal)

5. Client keeps sending **RopProgress** ROP requests, and server keeps sending **RopProgress** ROP response buffers telling client the current progress status.

Finally, when server has completed the the **RopEmptyFolder** ROP, instead of sending a **RopProgress** ROP response buffer, it sends the **RopEmptyFolder** ROP response buffer back. The following is the last **RopProgress** ROP request that the client makes.

0000: 50 00 00 00

RopId: 0x50 (**RopProgress**)

LogonId: 0x00

InputHandleIndex: 0x00

WantCancel: 0x00 (FALSE)

6. Server responds by sending the **RopEmptyFolder** ROP response buffer back, formatted as follows.

0000: 58 00 00 00 00 00 00

RopId: 0x58 (**RopEmptyFolder**)

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

ParitalCompletion: 0x00 (FALSE)

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the Property and Stream Object Protocol. 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
- Microsoft Office Outlook 2003
- Microsoft Office Outlook 2007
- Microsoft Outlook 2010
- Microsoft Outlook 2013
- Microsoft Outlook 2016

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 2.2.9: Exchange 2003 and Exchange 2007 do not support Logon objects for the **RopQueryNamedProperties** ROP ([MS-OXCROPS] section 2.2.8.10).

<2> Section 2.2.10.1: Exchange 2003 and Exchange 2007 support the combination of the **Move** bit and the **NoOverwrite** bit in the **CopyFlags** field.

<3> Section 2.2.11.1: Exchange 2003 and Exchange 2007 support the combination of the **Move** bit and the **NoOverwrite** bit in the **CopyFlags** field.

<4> Section 3.2.5.1: Exchange 2003 and Exchange 2007 do not ignore the **PropertySizeLimit** field. When the property is a **PtypBinary** type, a **PtypObject** type, or a string property, Exchange 2003 and Exchange 2007 return the **PtypErrorCode** type with a value of NotEnoughMemory (0x8007000E) in place of the property value if the value is larger than either the available space in the ROP response buffer or the size specified in the **PropertySizeLimit** field of the ROP request buffer. For information about the property types, see [MS-OXCDATA] section 2.11.1. For information about property error codes, see [MS-OXCDATA] section 2.4.2.

<5> Section 3.2.5.2: Exchange 2010 fails the **EcDoRpcExt2** method with a return value of 0x0000047D.

<6> Section 3.2.5.7: Exchange 2003 and Exchange 2007 remove the properties from the source object.

<7> Section 3.2.5.7: Exchange 2003 and Exchange 2007 ignore invalid bits and do not return the InvalidParameter error code (0x80070057).

<8> Section 3.2.5.8: Exchange 2003, Exchange 2007, and Exchange 2010 delete the properties from the source object.

<9> Section 3.2.5.8: Exchange 2003 and Exchange 2007 ignore invalid bits and do not return the InvalidParameter error code (0x80070057).

<10> Section 3.2.5.8: Exchange 2003, Exchange 2007, and Exchange 2010 do not return a NotSupported error (0x80040102) if the RopCopyTo ROP request specifies a public folder as either the source object or the destination object.

<11> Section 3.2.5.11: Exchange 2003, Exchange 2007, and Exchange 2010 do not return a NotSupported error (0x80040102) if the client attempts to open a stream on a public folder in any mode other than ReadOnly.

<12> Section 3.2.5.13: Exchange 2003 and Exchange 2007 return the StreamSizeError error code (0x80030070) if they write less than the amount requested.

<13> Section 3.2.5.18: The initial release version of Exchange 2010 does not implement the **RopCopyToStream** ROP ([MS-OXCROPS] section 2.2.9.8).

<14> Section 3.2.5.19: The initial release version of Exchange 2010 does not implement the **RopProgress** ROP ([MS-OXCROPS] section 2.2.8.13).

<15> Section 3.2.5.20: Exchange 2003 and Exchange 2007 implement the **RopLockRegionStream** ROP ([MS-OXCROPS] section 2.2.9.9).

<16> Section 3.2.5.21: Exchange 2003 and Exchange 2007 implement the **RopUnlockRegionStream** ROP ([MS-OXCROPS] section 2.2.9.10).

<17> Section 3.2.5.22: Exchange 2003 and Exchange 2007 implement the **RopWriteAndCommitStream** ROP ([MS-OXCROPS] section 2.2.9.11.

<18> Section 3.2.5.23: Exchange 2003 and Exchange 2007 implement the **RopCloneStream** ROP ([MS-OXCROPS] section 2.2.9.12).

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

N

O

P

R

S

T

V

W