[MS-OXCPRPT]:
Property and Stream Object Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Updated references.
12/03/2008	1.03		Updated IP notice
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	4.1.0	Minor	Updated the technical content.
05/05/2010	4.1.1	Editorial	Revised and edited the technical content.
08/04/2010	4.2	Minor	Clarified the meaning of the technical content.
11/03/2010	5.0	Major	Significantly changed the technical content.
03/18/2011	5.0	No change	No changes to the meaning, language, and formatting of the technical content.
08/05/2011	5.1	Minor	Clarified the meaning of the technical content.
10/07/2011	6.0	Major	Significantly changed 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 1.8, 2, and 3 of this specification are normative and contain RFC 2119 language. Sections 1.5 and 1.9 are also normative but cannot contain RFC 2119 language. All other sections and examples in this specification are informative.

1.1 Glossary

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

code page
Coordinated Universal Time (UTC)
GUID
handle
little-endian
remote procedure call (RPC)
Unicode

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

address book container
Address Book object
Attachment object
attachments table
distribution list
Folder object
global identifier
Logon object
long ID (LID)
mail user
Message object
multibyte character set (MBCS)
named property
property ID
property name
property tag
property type
recipient table
remote operation (ROP)
ROP buffer
ROP request
ROP request buffer
ROP response buffer
Server object
Server object handle
Server object handle table
store
Store object
Stream object
string property
tagged property

The following terms are specific to this document:

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

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

[MS-OXCFOLD] Microsoft Corporation, "Folder Object Protocol Specification".

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

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

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

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

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

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

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

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

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

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. Valid data types for properties are described in [MS-OXCDATA] section 2.11.1. All of the properties that are defined for objects are listed in [MS-OXPROPS] section 2.

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 **property tag**, which comprises the property ID and the **property type**, is used to specify a particular property. A client can obtain a property ID for a named property by querying the server for the property name-to-property ID mapping. A query for the reverse mapping (property ID-to-property name) allows the client to obtain the **property name** from the property ID.

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.

There are two different models for saving changes of 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 Wire Format Protocol, described in [MS-OXCRPC], to transmit the ROP buffers between client and server.
- The the Store Object Protocol, described in [MS-OXCSTOR], to log on to the **store**.

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

1.5 Prerequisites/Preconditions

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

1.6 Applicability Statement

This protocol is applicable when a client needs to read, modify, or delete properties that are associated with a Server object.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

A client can create its own named properties for an object. A named property has a dynamic property ID that is assigned by the server. A mapping of property names 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 are received from the server using the underlying remote procedure call (RPC) transport as specified in [MS-OXCROPS] section 2.1 and [MS-OXCRPC] section 2.1.

2.2 Message Syntax

2.2.1 Common Object Properties

The properties specified in this section are present on all objects. When a property is specified as "read-only for the client", any request to change the value of that property MUST be ignored by the server and SHOULD<1> result in an error.

2.2.1.1 PidTagAccess Property

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

The **PidTagAccess** property ([MS-OXPROPS] section 2.576) 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 2.578) indicates the client's access level to the object. This property does not apply to Folder objects and Logon objects. This property is read-only for the client. MUST be one of the following values:

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 **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 2.823) 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 2.866) indicates the type of Server object. This property does not apply to Folder objects and Logon objects. This property is read-only for the client. The value of this property MUST be one of the following values:

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 2.962) 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 2.1047) 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. Objects that are supported for this operation are: 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

2.2.2.1.1 PropertySizeLimit

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 **PropertySizeLimit**.

2.2.2.1.2 WantUnicode

If nonzero, indicates that string properties that are requested with **PtypUnspecified** ([MS-OXCDATA] section 2.11.1) as the property type are 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 encoded in **multibyte character set (MBCS)** format.

2.2.2.1.3 PropertyTagCount

Specifies the number of the property tags contained in the **PropertyTags** field.

2.2.2.1.4 PropertyTags

An array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). This is a list of property tags for which the client is requesting the value.

2.2.2.2 RopGetPropertiesSpecific ROP Response Buffer

2.2.2.2.1 RowData

The **RowData** field contains 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. Objects that are supported for this operation are: 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

2.2.3.1.1 PropertySizeLimit

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 **PropertySizeLimit**.

2.2.3.1.2 WantUnicode

If nonzero, indicates that string properties that are requested with **PtypUnspecified** ([MS-OXCDATA] section 2.11.1) as the property type are 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 encoded in multibyte character set (MBCS) format.

2.2.3.2 RopGetPropertiesAll ROP Response Buffer

2.2.3.2.1 PropertyValueCount

Specifies the number of elements in the **PropertyValues** field.

2.2.3.2.2 PropertyValues

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. 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. Objects that are supported for this operation are: 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

None.

2.2.4.2 RopGetPropertiesList ROP Response Buffer

2.2.4.2.1 PropertyTagCount

Specifies the number of property tag contained in the **PropertyTags** field.

2.2.4.2.2 PropertyTags

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 propertieson an object. Objects that are supported for this operation are: 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

2.2.5.1.1 PropertyValueSize

Specifies the number of bytes in **PropertyValueCount** field plus the number of bytes in the **PropertyValues** field.

2.2.5.1.2 PropertyValueCount

Specifies the number of elements contained in the **PropertyValues** field.

2.2.5.1.3 PropertyValues

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

2.2.5.2.1 PropertyProblemCount

Specifies the number of elements contained in the **PropertyProblems** field.

2.2.5.2.2 PropertyProblems

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 **RopSetProperties** ([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, **RopSetPropertiesNoReplicate** works exactly the same way as **RopSetProperties**.

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 RopDeleteProperties ROP

The **RopDeleteProperties** ROP ([MS-OXCROPS] section 2.2.8.8) removes the the specified properties from an object. Objects that are supported for this operation are: 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

2.2.7.1.1 PropertyTagCount

Specifies the number of property tags contained in the **PropertyTags** field.

2.2.7.1.2 PropertyTags

An array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). This is a list of property tags for the properties that the client is deleting. .

2.2.7.2 RopDeleteProperties ROP Response Buffer

2.2.7.2.1 PropertyProblemCount

Specifies the number of elements contained in the **PropertyProblems** field.

2.2.7.2.2 PropertyProblems

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 **RopDeleteProperties** ([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, **RopDeletePropertiesNoReplicate** works exactly 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].

2.2.9 RopQueryNamedProperties ROP

The **RopQueryNamedProperties** ROP ([MS-OXCROPS] section 2.2.8.10) queries an object for all the named properties. Objects that are supported for this operation are: Message objects, Folder objects, and Attachment objects.<2>

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

2.2.9.1 RopQueryNamedProperties ROP Request Buffer

2.2.9.1.1 QueryFlags

This field contains bits that control which type of named property is returned.The valid bits for this field are listed in the following table.

Name	Value	Description
NoStrings	0x01	Named properties with a **Kind** ([MS-OXCDATA] section 2.6.1) of 0x01 MUST NOT be included in the response.
NoIds	0x02	Named properties with a **Kind** of 0x00 MUST NOT be included in the response.

2.2.9.1.2 HasGUID

If the **HasGUID** field is nonzero then the **PropertyGUID** field MUST be included in the request. If the **HasGUID** field is zero then the **PropertyGUID** field MUST NOT be present.

2.2.9.1.3 PropertyGUID

If this field is present, only named properties with **GUIDs** matching the value of **PropertyGUID** are returned in a successful response.

2.2.9.2 RopQueryNamedProperties ROP Response Buffer

2.2.9.2.1 IdCount

Specifies the number of elements contained in the **PropertyIds** field.

2.2.9.2.2 PropertyIds

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.

2.2.9.2.3 PropertyNames

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. Objects that are supported for this operation are: Folder objects, Attachment objects, and Message objects. Also, the source and destination object MUST be of the same type.

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

2.2.10.1.1 WantAsynchronous

If this field is set to zero, then the ROP is processed synchronously. If this field is set to nonzero, then the ROP is processed either synchronously or asynchronously.

2.2.10.1.2 CopyFlags

This field contains bits that control options for moving or copying properties. The valid bits are listed in the following table.<3>

Name	Value	Description
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.

2.2.10.1.3 PropertyTagCount

Specifies the number of elements contained in the **PropertyTags** field.

2.2.10.1.4 PropertyTags

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

2.2.10.2.1 PropertyProblemCount

Specifies the number of elements contained in the **PropertyProblems** field. This field MUST NOT be present if the **ReturnValue** field is set to **NullDestinationObject**.

2.2.10.2.2 PropertyProblems

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**.

2.2.10.2.3 DestHandleIndex

The **DestHandleIndex** field MUST be present in a response if the **ReturnValue** field is set to **NullDestinationObject**, and it MUST be set to the value of the **DestHandleIndex** field of the request buffer. The **DestHandleIndex** field MUST NOT be present in a response if the **ReturnValue** field is not set to **NullDestinationObject**.

2.2.11 RopCopyTo ROP

The **RopCopyTo** ROP ([MS-OXCROPS] section 2.2.8.12) ROP is used to copy or move all but a specified few properties from a source object to a destination object. Objects that are supported for this operation are: 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

2.2.11.1.1 WantAsynchronous

If this field is set to zero, then this ROP is processed synchronously. If this field is set to nonzero, then this ROP is processed either synchronously or asynchronously.

2.2.11.1.2 WantSubObjects

If this field isset to nonzero, then sub-objects MUST also be copied. Otherwise they are not.

2.2.11.1.3 CopyFlags

This field contains bits that control options for moving or copying properties. The valid bits are listed in the following table.<4>

Name	Value	Description
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.

2.2.11.1.4 ExcludedTagCount

Specifies the number of elements contained in the **ExcludedTags** field.

2.2.11.1.5 ExcludedTags

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

2.2.11.2.1 PropertyProblemCount

Specifies the number of elements contained in the **PropertyProblems** field. This field MUST NOT be present if the **ReturnValue** field is set to **NullDestinationObject**.

2.2.11.2.2 PropertyProblems

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**.

2.2.11.2.3 DestHandleIndex

The **DestHandleIndex** field MUST be present in a response if the **ReturnValue** field is set to **NullDestinationObject**, and it MUST be set to the value of the **DestHandleIndex** field of the request buffer. The **DestHandleIndex** field MUST NOT be present in a response if the **ReturnValue** field is not set to **NullDestinationObject**.

2.2.12 RopGetPropertyIdsFromNames ROP

The **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1) maps abstract, client-defined property names into concrete 16-bit property IDs (15 of which are significant) and registers new property IDs. Objects that are supported for this operation are: 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

2.2.12.1.1 Flags

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; otherwise, this field is set to 0x00.

2.2.12.1.2 PropertyNameCount

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.

2.2.12.1.3 PropertyNames

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

2.2.12.2 RopGetPropertyIdsFromNames ROP Response Buffer

2.2.12.2.1 PropertyIdCount

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.

2.2.12.2.2 PropertyIds

This field is an array of 16-bit integers. Each integer is a property ID that is mapped from a property name that is specified in the **PropertyNames** field of the ROP request buffer. For property names that 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 property names 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 **Kind** ([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 property names.

Objects that are supported for this operation are: Message objects, Attachment objects, Folder objects and Logon objects. Named property IDs 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 abstract name for such IDs is comprised of the **PS-MAPI** ([MS-OXPROPS] section 1.3.2) namespace GUID, a **Kind** ([MS-OXCDATA] section 2.6.1) value equal to 0x00, and a **LID** ([MS-OXCDATA] section 2.6.1) value equal to the property ID given.

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

2.2.13.1.1 PropertyIdCount

Specifies the number of 16-bit property IDs contained in the **PropertyIds** field.

2.2.13.1.2 PropertyIds

An array of 16-bit integers. Each integer is a property ID to be mapped to a property name. 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

2.2.13.2.1 PropertyNameCount

Specifies the number of elements contained in the **PropertyNames** field. This value MUST be equal to the value of the **PropertyIdCount** field of the ROP request buffer.

2.2.13.2.2 PropertyNames

An array of **PropertyName** structures ([MS-OXCDATA] section 2.6). Each structure specifies a property name 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. Objects that are supported for this operation are: Folder objects, Attachment objects, and Message objects. Only single-valued **PtypBinary**, **PtypObject**, **PtypString8**, and **PtypString** type properties are supported for Attachment objects and Message objects. Only single-valued **PtypBinary** type properties are supported for 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].

2.2.14.1 RopOpenStream ROP Request Buffer

2.2.14.1.1 PropertyTag

The **PropertyTag** field contains a property tag that specifies which property the client is opening. The format of a property tag is specified in ([MS-OXCDATA] section 2.9).

2.2.14.1.2 OpenModeFlags

Specifies the mode for opening the stream. The **OpenModeFlags** field MUST be set to one of the following values.

Name	Value	Description
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 stream for read/write access. This mode is required for a property that has not been set.
BestAccess	0x03	If the object this ROP is acting on was opened with read/write access, then the stream MUST be opened with read/write access. Otherwise, the stream MUST be opened with read-only access.

2.2.14.2 RopOpenStream ROP Response Buffer

2.2.14.2.1 StreamSize

Specifiesthe number of bytes in the 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 ROP is supported only for 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

2.2.15.1.1 ByteCount

Specifies 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 **MasimumByteCount** field.

2.2.15.1.2 MaximumByteCount

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

2.2.15.2.1 DataSize

Specifies the number of bytes in the **Data** field.

2.2.15.2.2 Data

Contains 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 ROP is supported only for 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

2.2.16.1.1 DataSize

Specifies the number of bytes in the **Data** field.

2.2.16.1.2 Data

Contains the data to be written to the stream.

2.2.16.2 RopWriteStream ROP Response Buffer

2.2.16.2.1 WrittenSize

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. This ROP is supported only for 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

None.

2.2.17.2 RopCommitStream ROP Response Buffer

None.

2.2.18 RopGetStreamSize ROP

The **RopGetStreamSize** ROP ([MS-OXCROPS] section 2.2.9.5) retrieves the size of the stream. This ROP is supported only for 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

None.

2.2.18.2 RopGetStreamSize ROP Response Buffer

2.2.18.2.1 StreamSize

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 ROP is supported only for 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

2.2.19.1.1 StreamSize

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

None.

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 ROP is supported only for 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

2.2.20.1.1 Origin

Specifies the point of origin for the seek pointer. The **Origin** field MUST be set to one of the following values:

Value	Description
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.

2.2.20.1.2 Offset

Specifies the number of bytes that the seek pointer is to be offset from the origin. This value can be positive or negative.

2.2.20.2 RopSeekStream ROP Response Buffer

2.2.20.2.1 NewPosition

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 ROP is supported only for 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

2.2.21.1.1 ByteCount

Specifies the number of bytes to copy from the source stream to the destination stream.

2.2.21.2 RopCopyToStream ROP Response Buffer

2.2.21.2.1 ReadByteCount

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.

2.2.21.2.2 WrittenByteCount

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.

2.2.21.2.3 DestHandleIndex

The **DestHandleIndex** field MUST be present in a response if the **ReturnValue** field is set to **NullDestinationObject**, and it MUST be set to the value of the **DestHandleIndex** field that is specified in the ROP request buffer. The **DestHandleIndex** MUST NOT be present in a response if the **ReturnValue** field is not set to **NullDestinationObject**.

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.

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

2.2.22.1.1 WantCancel

This field is set to zero if the client wants the current operation to continue. If the value is nonzero, the client is requesting that the server attempt to cancel the operation.

2.2.22.2 RopProgress ROP Response Buffer

2.2.22.2.1 CompletedTaskCount

Specifiesthe approximate number of tasks that the server has completed.

2.2.22.2.2 TotalTaskCount

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 ROP is supported only for 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

2.2.23.1.1 RegionOffset

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.

2.2.23.1.2 RegionSize

An unsigned 64-bit integer that specifies the size, in bytes, of the region to be locked.

2.2.23.1.3 LockFlags

Controls the read/write access to the locked region. The **LockFlags** field MUST be set to one of the following values.

Value	Description
0x00000001	If this lock is granted, then 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, then 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

None.

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 ROP is supported only for 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

2.2.24.1.1 RegionOffset

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.

2.2.24.1.2 RegionSize

An unsigned 64-bit integer that specifies the size, in bytes, of the region to be unlocked.

2.2.24.1.3 LockFlags

The value of the **LockFlags** field MUST be one of the values specified in section 2.2.23.1.3. 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 to be unlocked.

2.2.24.2 RopUnlockRegionStream ROP Response Buffer

None.

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 ROP is supported only for 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

2.2.25.1.1 DataSize

Specifies the number of bytes in the **Data** field.

2.2.25.1.2 Data

Contains the data to be written and committed to the stream.

2.2.25.2 RopWriteAndCommitStream ROP Response Buffer

2.2.25.2.1 WrittenSize

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 ROP is supported only for 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

None.

2.2.26.2 RopCloneStream ROP Response Buffer

None.

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** request ([MS-OXCROPS] section 2.2.6.3). Changes to properties of an Attachment object are not saved until the client sends a **RopSaveChangesAttachment** request ([MS-OXCROPS] section 2.2.6.15) followed by a **RopSaveChangesMessage** 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 not persisted until the client sends **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4) on 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 **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1) to map property names 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 property name 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 Reading a Property

If any of the properties to be read are named properties, the client uses **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1) to obtain the property IDs for those properties.

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

3.1.4.2 Setting a Property

If any of the properties to be set are named properties, the client uses **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1) to obtain the property IDs for those properties

The client then uses **RopSetProperties** ([MS-OXCROPS] section 2.2.8.6) to set the properties. 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 theROP 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.

3.1.4.3 Reading a Property as a Stream

If the server returned **NotEnoughMemory** (0x8007000E) in place of the property value in the ROP response buffer of **RopGetPropertiesSpecific** or **RopGetPropertiesAll**, 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 uses **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1) to obtain the property ID for that property.

The client then uses **RopOpenStream** to obtain a handle to a Stream object on the property. The client sets the **OpenModeFlags** field of the ROP request buffer to **ReadOnly** if the client is just going to read from the stream. If the client wishes to write to the stream, the client sets the **OpenModeFlags** field to **BestAccess** or **ReadWrite**. The client checks the **ReturnValue** field of the ROP response buffer to verify that the handle was retrieved.

If the client does not want to read from the start of the stream, the client uses **RopSeekStream** ([MS-OXCROPS] section 2.2.9.7) to set the seek pointer. The client then sends one or more **RopReadStream** requests ([MS-OXCROPS] section 2.2.9.2) to read data from the stream. 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 issue another **RopReadStream** request to determine if there is more data in the stream. 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.

The client uses **RopRelease** ([MS-OXCROPS] section 2.2.15.3) after it is done with the Stream object.

3.1.4.4 Setting a Property with a Stream

If the property to be set is a named property, the client uses **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1) to obtain the property ID for that property.

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

If the client does not want to write from the start of the stream, the client uses **RopSeekStream** ([MS-OXCROPS] section 2.2.9.7) to set the seek pointer. The client then uses one or more **RopWriteStream** requests ([MS-OXCROPS] section 2.2.9.3) to write data to the stream.

The client uses **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4) to save the changes to the property of a Folder object. The client uses **RopSaveChangesMessage** ([MS-OXCROPS] section 2.2.6.3) to save the changes to the property of a Message object. The client uses

RopSaveChangesAttachment ([MS-OXCROPS] section 2.2.6.15) followed by **RopSaveChangesMessage** to save the changes to the property of an Attachment object.

The client uses **RopRelease** ([MS-OXCROPS] section 2.2.15.3) once it is done with the Stream object.

3.1.5 Message Processing Events and Sequencing Rules

With the exception of **RopProgress** ([MS-OXCROPS] section 2.2.8.13), all the messages specified in section 2 of this protocol are all 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 ofr any of the following ROPs, the server can respond with a **RopProgress** response.

- **RopCopyTo** (section 2.2.11)
- **RopCopyProperties** (section 2.2.10)
- **RopSetReadFlags** ([MS-OXCMSG] section 2.2.3.10)
- **RopMoveCopyMessages** ([MS-OXCFOLD] section 2.2.1.6)
- **RopMoveFolder** ([MS-OXCFOLD] section 2.2.1.7)
- **RopCopyFolder** ([MS-OXCFOLD] section 2.2.1.8)
- **RopHardDeleteMessagesAndSubfolders** ([MS-OXCFOLD] section 2.2.1.10)
- **RopEmptyFolder** ([MS-OXCFOLD] section 2.2.1.9)
- **RopDeleteMessages** ([MS-OXCFOLD] section 2.2.1.11)
- **RopHardDeleteMessages** ([MS-OXCFOLD] section 2.2.1.12)

The server can respond with a **RopProgress** ROP response buffer to notify the client of its current progress. 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 send additional **RopProgress** requests to request additional status, or to abort the operation. In response to the client's **RopProgress** request, the server MUST respond with either the response to the original ROP, or another **RopProgress** 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. See 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 all properties on the object, including those set by any client, server, or computed properties. 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 of the ROP request buffer.

If the **WantUnicode** field is set to a nonzero value, the server MUST return string properties which are requested without a specified type (**PtypUnspecified**) in Unicode format. If the **WantUnicode** field is set to zero, the server MUST return string properties which 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.

When the property is a **PtypBinary** type, a **PtypObject** type, or a **string property**, the server SHOULD<5> 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 details about the property types, see [MS-OXCDATA] section 2.11.1. For details about property error codes, see [MS-OXCDATA] section 2.4.2.

This ROP is supported for Message objects, Folder objects, Attachment objects, and Logon objects.

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. 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) 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.

This ROP is supported for Message objects, Folder objects, Attachment objects and Logon objects.

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 property specified in the **PropertyValues** field of the ROP request buffer. If the server fails to modify 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.

For Message objects, the new value of the properties MUST be made available immediately for retrieval by an ROP that uses the same Message object handle. For example, if the client uses the same object handle in a **RopGetPropertiesAll** 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** ([MS-OXCROPS] section 2.2.6.3) is issued.

For Attachment objects, the new value of the properties MUST be made available immediately for retrieval by an ROP that uses the same Attachment object handle. However, the modified value MUST NOT be persisted to the database until a successful **RopSaveChangesAttachment** ([MS-OXCROPS] section 2.2.6.15) followed by a successful **RopSaveChangesMessage** 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.

This ROP is supported for Message objects, Folder objects, Attachment objects and Logon objects.

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 remove the value of 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 **NotFound** ([MS-OXCDATA] section 2.4.2) in place of a value.

For Message objects, the value of 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 **RopGetPropertiesSpecific** ([MS-OXCROPS] section 2.2.8.3) or **RopGetPropertiesAll** ([MS-OXCROPS] section 2.2.8.4), the values MUST be deleted. However, the deleted values MUST NOT be persisted to the database until a successful **RopSaveChangesMessage** ([MS-OXCROPS] section 2.2.6.3) is issued.

For Attachment objects, the value of the properties MUST be removed immediately when using the same handle. However, the deleted values 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 value of the properties MUST be removed immediately without requiring another ROP to commit the change.

This ROP is supported for Message objects, Folder objects, Attachment objects and Logon objects.

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 (section 2.2.9.1.1), 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 (section 2.2.9.1.3) 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.

This ROP is supported for Message objects, Folder objects, and Attachment objects.

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 SHOULD<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.

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.

If the client requests asynchronous execution, then the server can execute this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([MS-OXCROPS] section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopCopyProperties** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **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.

This ROP is supported for Message objects, Attachment objects, and Folder objects.

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.

This ROP works in the same way as **RopCopyProperties** ([MS-OXCROPS] section 2.2.8.11) except that it MUST copy or move all writable non-excluded properties from the source object to the destination object.

If the client requests asynchronous execution, then the server can execute this ROP asynchronously. During asynchronous processing, the server can return a **RopProgress** ROP response buffer ([MS-OXCROPS] section 2.2.8.13) to indicate that the operation is still processing, or it can return a **RopCopyTo** ROP response buffer to indicate that the operation has already completed. If the operation fails at any point during the asynchronous processing, the server returns a **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.

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 sub-object cannot be copied because there is already a sub-object existing in the destination object with the same display name (**PidTagDisplayName**, as defined in [MS-OXPROPS] section 2.734) as the sub-object 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**, the server MUST return the **PropertyName** associated with the **PropertyId**.

3. For property ID that do not have an associated **PropertyName**, the associated name MUST be composed:

   - A **Kind** of 0xFF. There is no other return data for this entry.

This ROP is supported for Folder objects, Message objects, Attachment objects and Logon objects.

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** ([MS-OXCROPS] section 2.2.8.1) is acting on a Logon object, the server MUST enumerate all property IDs associated with property names. 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) specifies the **PS_MAPI** property set, the returned property ID is obtained from the **LID** ([MS-OXCDATA] section 2.6.1). 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 **Name** value to all lowercase. Property sets are specified in [MS-OXPROPS] section 1.3.2.
2. Find the property ID associated with the property names that precisely match the **Kind** and identifier values from the name entry. The identifier value is the value of **Name** if **Kind** is equal to 0x01 or is the value of **LID** if **Kind** is equal to 0x00.
3. For unfound rows, the returned property ID MUST be 0x0000 unless the 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 3 are all met, a new property ID is associated with this property name. The newly assigned property ID MUST be unique in that it MUST NOT be assigned to another property name, and MUST NOT be equal to 0xFFFF. The newly assigned property ID MUST be greater than 0x8000.

Because there are only 32,767 available property IDs (15 significant bits), the server MUST impose a limit of at most 32,767 on the total number of mappings it will allow before returning errors to the client. If the server reaches this limit, then it MUST return an **OutOfMemory** error in response to a request to create further mappings.

This ROP is supported for Folder objects, Message objects, Attachment objects and Logon objects.

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.

When the client sends the server a **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1) request, the server MUST parse the request as specified in [MS-OXCROPS] section 3.2.5.1 and section 2.2.14.1. The server MUST respond with a **RopOpenStream** response as specified in section 2.2.14.2.

The server MUST open the stream in the mode indicated by the **OpenModeFlags** field as specified by the table in section 2.2.14.1.2.

The implementation of **RopOpenStream** 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. The client MUST call **RopRelease** ([MS-OXCROPS] section 2.2.15.3) on the Folder object to free this resource.

The server MUST store the location of the seek pointer until the client calls **RopRelease**. 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.

This ROP is supported for Message objects, Attachment objects and Folder objects.

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

Name	Value	Description
NotFound	0x8004010F	The property tag does not exist for the object and it cannot be created because Create was not specified in *OpenModeFlags*.

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.

This ROP is supported only for Stream objects.

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<8> return **StreamSizeError** 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 was 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.

This ROP is supported only for Stream objects.

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

Name	Value	Description
StreamSizeError	0x80030070	The write will exceed the maximum stream size.
TooBig	0x80040305	The result set of the operation is too big for the server to return.
STG_E_ACCESSDENIED	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** request ([MS-OXCROPS] section 2.2.9.1) with the data from the Stream object.

This ROP is supported only for Stream objects.

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.

This ROP is supported only for Stream objects.

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.

This ROP is supported only for Stream objects.

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 **StreamSeekError** 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.

This ROP is supported only for Stream objects.

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

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 *Origin* 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.

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 was copied.

This ROP is supported only for Stream objects.

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

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.

If the server is not done with the asynchronous operation, then it MUST respond with a **RopProgress** response.<9> If the server recieves a a **RopProgress** 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 SHOULD <10> 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 **AccessDenied** (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 **NetworkError** (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 (section 2.2.23.1.3).

This ROP is supported only for Stream objects.

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 SHOULD <11> 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.

This ROP is supported only for Stream objects.

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 SHOULD<12> process this ROP as follows:

1. Process the request as specified in section 3.2.5.13.

2. Process the request as specified in section 3.2.5.14.

This ROP is supported only for Stream objects.

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 SHOULD <13> 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.

This ROP is supported only for Stream objects.

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 which gets transmitted over the wire. Also note that the data for a multi-byte 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 packed in one or more **RPC** calls according to the description in [MS-OXCRPC] section 3. These examples assume the client has already successfully logged onto 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 seriesis 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**)

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** with a value of 10 because the type of the **PidTagRuleSequence** property ([MS-OXPROPS] section 2.1012) is **PtypInteger32**.

4.1 Getting PropertyIds

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

4.1.1 Client Request Buffer

In this example the client calls server to get property IDs for 2 named properties it wants to 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 **RopGetPropertyIdsFromNames** ([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 property names that do not already have an existing mapping.

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

The remaining bytes form the **PropertyNames** field. Each row in **PropertyNames** 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 Name 1:

Kind: 0x01 MNID_STRING

GUID: 00062002-0000-0000-c000-000000000046

NameSize: 0x14

Name: TestProp1

Property Name 2:

Kind: 0x01 MNID_STRING

GUID: 00062002-0000-0000-c000-000000000046

NameSize: 0x14

Name: TestProp2

4.1.2 Server Response to Client Request

```
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**)

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

The next two bytes is specify the number of property IDs that are packed in the buffer as described in section 2.2.12.2.1.

PropertyIdCount: 0x0002 (2 properties in the property tag array)

The remaining bytes are for the **PropertyIds** field where each entry in the array is four-byte property ID as described in section 2.2.12.2.2.

```
0009: 3e 86 3f 86
```

property ID: 0x863e

property ID: 0x863f

4.2 Setting Properties

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

4.2.1 Client Request Buffer

In this example, the client is setting values for 2 properties on a Message object.

A complete ROP request buffer is a variable length sequence, with seven required bytes and followed by 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 **RopSetProperties** ([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 **RopSetProperties** 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. Size of Count and Array that follows.

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

The remaining bytes constitute the **PropertyValues** field, where each row in 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 to Client Request

```
0000: 0a 00 00 00 00 00 00 00
```

The first six bytes of the ROP response buffer 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 is the **PropertyProblemCount** field described in section section 2.2.5.2.1.

```
0006: 00 00
```

PropertyProblemCount: 0x0000. Count of Problem property that follows. 0 indicates that all properties were set successfully.

4.3 Getting Properties

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

4.3.1 Client Request Buffer

In this example, the client is requesting property values for specific properties from the server. The first two are named properties and the third one is standard property. The property IDs for the named properties were gotten from the server by calling **RopGetPropertyIdsFromNames**. For more details, see section 4.1 .

A complete ROP request buffer 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**)

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 prop size set; maximum limit is default

WantUnicode: 0x0001. Nonzero means Unicode

PropertyTagCount: 0x0003. 3 properties tags in array

The remaining bytes constitute the **PropertyTags** field as described in section 2.2.2.1.4.

```
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 to Client Request

```
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 ROP response buffer contain the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

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

RopId: 0x07 (**RopGetPropertiesSpecific**)

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

The following sections give examples of how to set value for a property using streams. The ROPs covered are the following:

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

4.4.1.1 Client Request Buffer

A complete ROP request buffer is nine bytes in length. An example of the request buffer 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 **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1).

```
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**)

OpenModeFlags: 0x01. ReadWrite Mode

4.4.1.2 Server Response to Client Request

```
0000: 2b 01 00 00 00 00 15 2e-00 00
```

The first six bytes of the ROP response buffer 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 is 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 following example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3) operation as described in section 2.2.16.

4.4.2.1 Client Request Buffer

A complete ROP request buffer 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 **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3).

```
0000: 2d 01 01
```

RopId: 0x2d (**RopWriteStream**)

LogonId: 0x01

InputHandleIndex: 0x01

The next two bytes in the ROP request buffer is the **DataSize** field as described in section 2.2.16.1.1.

```
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 to Client Request

```
0000: 2d 01 00 00 00 00 15 2e
```

The first six bytes of the ROP response buffer 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 2.2.16.2.1.

```
0006: 15 2e
```

WrittenSize: 0x2e15 (11797)

4.4.3 Committing a Stream

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

4.4.3.1 Client Request Buffer

For the purposes of this example, it is assumed that there is a stream open before this ROP is called. A complete ROP request buffer for **RopCommitStream** ([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 to Client Request

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

The six bytes of the ROP response buffer contain the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

RopId: 0x5d (**RopCommitStream**)

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

4.5 Asynchronous Progress

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

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

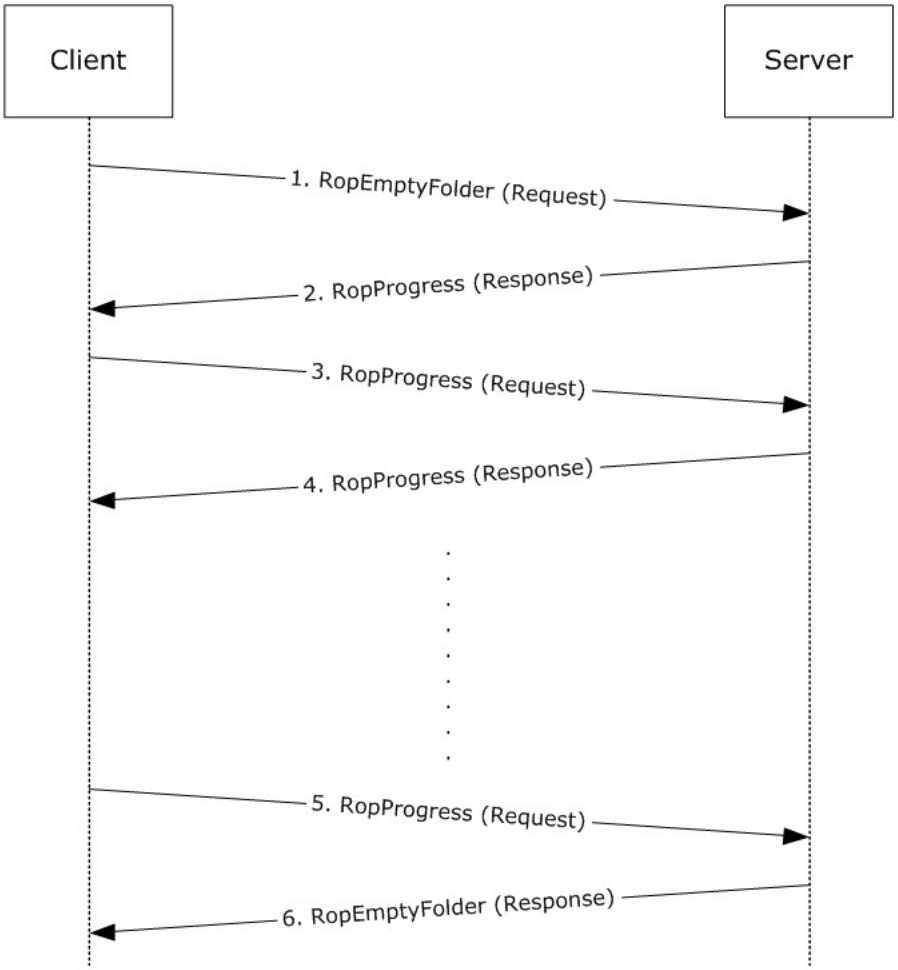

Figure 1: Sequence in which ROP are passed between client and server

1. Client sends a **RopEmptyFolder** 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 **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** request buffer asking server for the 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 **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** requests and server keeps sending **RopProgress** ROP response buffers telling client the current progress status.

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

0000: 50 00 00 00

RopId: 0x50 (**RopProgress**)

LogonId: 0x00

InputHandleIndex: 0x00

WantCancel: 0x00 (FALSE)

6. Server responds by sending **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® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010

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

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

<1> Section 2.2.1: A request to change the value of a read-only property does not result in an error on Exchange 2010.

<2> Section 2.2.9: Exchange 2010 also supports Logon objects for the **RopQueryNamedProperties** ROP ([MS-OXCROPS] section 2.2.8.10).

<3> Section 2.2.10.1.2: Exchange 2010 does not support the combination of the **Move** flag and the **NoOverwrite** flag in the **CopyFlags** field, and will return **InvalidParameter** if both flags are set.

<4> Section 2.2.11.1.3: Exchange 2010 does not support the combination of the **Move** flag and the **NoOverwrite** flag in the **CopyFlags** field, and will return **InvalidParameter** if both flags are set.

<5> Section 3.2.5.1: Exchange 2010 ignores the **PropertySizeLimit** field and does not return the **PtypErrorCode** type with a value of **NotEnoughMemory** (0x8007000E) in place of the property value.

<6> Section 3.2.5.7: Exchange 2010 does not 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.

<8> Section 3.2.5.13: Exchange 2010 returns **TooBig** if it writes less than the amount requested.

<9> Section 3.2.5.19: The initial release version of Exchange 2010 does not implement **RopProgress** ([MS-OXCROPS] section 2.2.8.13).

<10> Section 3.2.5.20: Exchange 2010 does not implement this ROP.

<11> Section 3.2.5.21: Exchange 2010 does not implement this ROP.

<12> Section 3.2.5.22: Exchange 2010 does not implement this ROP.

<13> Section 3.2.5.23: Exchange 2010 does not implement this ROP.

7 Change Tracking

This section identifies changes that were made to the [MS-OXCPRPT] protocol document between the August 2011 and October 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1 Introduction	Added information about which sections of the specification are normative and can contain RFC 2119 language.	Y	New content added for template compliance.
1.8 Vendor-Extensible Fields	Moved content from the "Named Properties" child section.	N	Content updated.
2.2.1 Common Object Properties	Added product behavior note specifying that a request to change a read-only property does not result in an error on Exchange 2010.	N	New product behavior note added.
2.2.1.2 PidTagAccessLevel Property	Specified that the property does not apply to Folder objects or Logon objects.	N	Content updated.
2.2.1.5 PidTagLastModifierName Property	Specified that the property does not apply to Folder objects or Logon objects.	N	Content updated.
2.2.1.7 PidTagObjectType Property	Specified that the property does not apply to Folder objects or Logon objects.	N	Content updated.
2.2.1.8 PidTagRecordKey Property	Specified that the property does not apply to Folder objects or Logon objects.	N	Content updated.
2.2.1.9 PidTagSearchKey Property	Specified that the property does not apply to Folder objects or Logon objects.	N	Content updated.
2.2.9	Added product behavior notes specifying that	N	New product

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
RopQueryNamedProperties ROP	Exchange 2010 also supports Logon objects for RopQueryNamedProperties.		behavior note added.
2.2.9.2.3 PropertyNames	Added reference for PropertyName stucture.	N	Content updated.
2.2.11.1.3 CopyFlags	Added product behavior note specifying that Exchange 2010 does not support the combination of the Move flag and the NoOverwrite flag.	N	New product behavior note added.
2.2.12.2.2 PropertyIds	Removed the statement about the ecWarnWithErrors (0x00040380) return code.	N	Content removed.
2.2.14 RopOpenStream ROP	Removed "Folder objects" from the list of valid objects for the ROP and deleted the statement "Folder objects only support PtypBinary type properties".	N	Content removed.
3.1.1 Abstract Data Model	Moved details about changes to a property, changes to a stream, and caching of property IDs from the "Property Transactions" section, the "Streams" section, and the "Named Properties" section.	N	Content updated.
3.2.5.1 Processing RopGetPropertiesSpecific	Clarified what code pages are used for strings in MBCS format.	N	Content updated.
3.2.5.1 Processing RopGetPropertiesSpecific	Reworded to specify that the server SHOULD return the PtypErrorCode type with a value of NotEnoughMemory.	N	Content updated.
3.2.5.1 Processing RopGetPropertiesSpecific	Added product behavior note to specify that Exchange 2010 ignores the PropertySizeLimit field.	N	New product behavior note added.
3.2.5.11 Processing RopOpenStream	Removed "Folder objects" from the list of objects that support the ROP.	N	Content removed.
3.2.5.13 Processing RopWriteStream	Revised details about how property values are persisted to the database for Message objects and Attachment objects.	Y	Content updated.
	Removed the "Property Transactions" section, the "Streams" section, and the "Named Properties" section from the "Abstract Data Model" section of the "Client Details".	N	Content removed.
	Removed the "Named Properties" section and moved the content to the "Vendor-Extensible Fields" section.	N	Content removed.

8 Index

S

T

V