[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.

Table of Contents

1 Introduction

This document specifies how to set, get, and delete data from a **property set**.

1.1 Glossary

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

code page
Coordinated Universal Time (UTC)
GUID
handle
little-endian
property set
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
long ID (LID)
mail user
Message object
named property
property ID
property name
property tag
property type
recipient table
remote operation (ROP)
ROP request buffer
ROP response buffer
rule
Server object
Server object handle table
Store object
string property
tagged property

The following terms are specific to this document:

property object: An abstract concept that specifies the shared behaviors of Message objects, Folder objects, Logon objects, and Attachment objects.

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-DTYP] Microsoft Corporation, "Windows Data Types".

[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

The Property and Stream Object Protocol is used to access and manage properties that are associated with an object. 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. An implementer 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. A client can also query the server for a complete list of property names.

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 transaction models for **Property objects**. 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

This protocol relies on understanding how **ROPs** are transmitted to the server using the underlying **remote procedure call (RPC)** transport. For more information, see [MS-OXCROPS] section 1.4 and [MS-OXCRPC] section 1.4).

1.5 Prerequisites/Preconditions

This protocol assumes the messaging client has previously logged on to the server and has acquired a **handle** to the property object on which it is going to operate. Methods to open the object and acquire a handle are dependent on the object type and are specified in detail for Message objects and Attachment objects in [MS-OXCMSG] section 3.1.4.1, for Folder objects in [MS-OXCFOLD] section 3.1.4.1, and for Logon objects in [MS-OXCSTOR] section 3.1.4.1.

1.6 Applicability Statement

The Property and Stream object protocol is used to query, modify, or delete properties associated with messaging objects on the server.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

1.8.1 Named Properties

Named properties allow clients to use additional property IDs without having to worry about reusing properties that other clients are using. Named properties provide a persistent mapping from property names to property IDs.

1.9 Standards Assignments

None.

2 Messages

Unless otherwise specified, sizes in this section are expressed in BYTEs.

The following data types are specified in [MS-DTYP] section 2:

BYTE

ULONG

The following data types are specified in [MS-OXCDATA] section 2.11.1:

PtypBinary

PtypBoolean

PtypErrorCode

PtypInteger32

PtypObject

PtypString

PtypString8

PtypTime

PtypUnspecified

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 Semantics

The following sections specify the usage of ROP requests and responses that are specific to the Property and Stream object protocol.

2.2.1 Common Object Properties

The properties specified in this section are present on all property 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 result in an error.

2.2.1.1 PidTagAccess

Type: **PtypInteger32**.

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. This property is described in [MS-OXPROPS] section 2.576.

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

Type: **PtypInteger32**.

Indicates the client's access level to the object. This property is read-only for the client. This property is described in [MS-OXPROPS] section 2.578. MUST be one of the following values:

Value	Meaning
0x00000000	Read-Only
0x00000001	Modify

2.2.1.3 PidTagChangeKey

Type: **PtypBinary**.

Contains a global identifier (GID) indicating the last change to the object ([MS-OXCFXICS] section 2.2.1.2.7). This property is read-only for clients. This property is described in [MS-OXPROPS] section 2.690.

2.2.1.4 PidTagCreationTime

Type: **PtypTime**

Contains the time the object was created in **UTC**. This property is read-only for clients. This property is described in [MS-OXPROPS] section 2.716.

2.2.1.5 PidTagLastModifierName

Type: **PtypString**.

Contains the name of the last **mail user** to modify the object. This property is read-only for clients. This property is described in [MS-OXPROPS] section 2.827.

2.2.1.6 PidTagLastModificationTime

Type: **PtypTime**.

Contains the time of the last modification to the object in UTC. This property is read-only for clients. This property is described in [MS-OXPROPS] section 2.825.

2.2.1.7 PidTagObjectType

Type: **PtypInteger32**.

Indicates the type of **Server object**. This property is read-only for the client. This property is described in [MS-OXPROPS] section 2.871. MUST be one of the following values:

Value	Meaning
0x00000001	**Store object**
0x00000002	**Address Book object**
0x00000003	Folder Object
0x00000004	**address book container**
0x00000005	Message object
0x00000006	mail user
0x00000007	Attachment object
0x00000008	**distribution list**

2.2.1.8 PidTagRecordKey

Type: **PtypBinary**.

Contains a unique binary-comparable identifier for a specific object. Whenever a copy of this object is created, the server generates a new value for the new object. This property is read-only for the client. This property is described in [MS-OXPROPS] section 2.968.

2.2.1.9 PidTagSearchKey

Type: **PtypBinary**.

Contains a unique binary-comparable key that identifies an object for a search. Whenever a copy of this object is created, this value is also copied from the original object. This property is read-only for clients. This property is described in [MS-OXPROPS] section 2.1058.

2.2.2 RopGetPropertiesSpecific

RopGetPropertiesSpecific ([MS-OXCROPS] section 2.2.8.3) queries for and returns the values of properties specified in the **PropertyTags** field. The property values MUST be returned in the same order as they were requested. Objects that are supported for this operation are: Message objects, Folder objects, Attachment objects and Logon objects.

2.2.2.1 Request Parameter Overview

2.2.2.1.1 PropertySizeLimit

Maximum size allowed for a property value. Properties larger than this *PropertySizeLimit* MUST get a **NotEnoughMemory** ([MS-OXCDATA] section 2.4.2) error returned for their value in the response. If this value is zero, the property values are only limited by the size of the ROP response buffer. If

this value is nonzero, the property values are limited by the size of the ROP response buffer and the value of *PropertySizeLimit*.

2.2.2.1.2 WantUnicode

If nonzero, indicates that string properties which are requested with **PtypUnspecified** ([MS-OXCDATA] section 2.11.1) are encoded by the server using the **Unicode** format. If *WantUnicode* is zero, the server returns string properties which are requested with **PtypUnspecified** as the property type using a multiple-byte character set.

Properties requested with a specific string type are returned using that type.

For properties on Message objects, the **code page** used for strings in MBCS format is the code page set on the Message object when it was opened if any, otherwise the code page of the Logon objects ([MS-OXCMSG]).

For properties on Attachment objects, the code page used for strings in MBCS format is the code page set on the parent Message object when it was opened if any otherwise the code page of the Logon objects.

For all other objects, the code page used for strings in MBCS format is the code page of the Logon objects.

2.2.2.1.3 PropertyTagCount

Count of the property tags in the **PropertyTags** field.

2.2.2.1.4 PropertyTags

Array of property tags. This is a list of property tags for which the client is requesting the value. The buffer format of a property tag is specified in [MS-OXCDATA] section 2.9.

2.2.2.2 Response Parameter Overview

2.2.2.2.1 RowData

The **RowData** field is an array of **PropertyRow** ([MS-OXCDATA] section 2.8.1) elements. The number of items in the array is specified in the **PropertyTagCount** field of the request buffer.

2.2.3 RopGetPropertiesAll

RopGetPropertiesAll ([MS-OXCROPS] section 2.2.8.4) queries for and returns all of the property tags and values that have been set. The server returns all properties necessary for the client to create an equivalent duplicate 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.

2.2.3.1 Request Parameter Overview

2.2.3.1.1 PropertySizeLimit

Maximum size allowed for a property value. Properties larger than this *PropertySizeLimit* MUST get a **NotEnoughMemory** ([MS-OXCDATA] section 2.4.2) error returned for their value in the response. If this value is zero, the property values are only limited by the size of the ROP response buffer. If

this value is nonzero, the property values are limited by the size of the ROP response buffer and the value of *PropertySizeLimit*.

2.2.3.1.2 WantUnicode

If nonzero, indicates that string properties which are requested with **PtypUnspecified** ([MS-OXCDATA] section 2.11.1) are encoded by the server using the Unicode format. If *WantUnicode* is zero, the server returns string properties which are requested with **PtypUnspecified** as the property type using a multiple-byte character set.

Properties requested with a specific string type are returned using that type.

For properties on Message objects, the code page used for strings in MBCS format is the code page set on the Message object when it was opened if any, otherwise the code page of the Logon objects.

For properties on Attachment objects, the code page used for strings in MBCS format is the code page set on the parent Message object when it was opened if any otherwise the code page of the Logon objects.

For all other objects, the code page used for strings in MBCS format is the code page of the Logon objects.

2.2.3.2 Response Parameter Overview

2.2.3.2.1 PropertyValueCount

The value of the *PropertyValueCount* field MUST be the number of elements in *PropertyValues*.

2.2.3.2.2 PropertyValues

The *PropertyValues* field is an array of **TaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.4) containing all property tags and property values on the queried object. If a value is larger than the available space in the ROP response buffer or if the property value is larger than *PropertySizeLimit* 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

RopGetPropertiesList ([MS-OXCROPS] section 2.2.8.5) queries for and returns all of the property tags that have been set. The server returns all properties necessary for the client to create an equivalent duplicate 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.

2.2.4.1 Request Parameter Overview

None.

2.2.4.2 Response Parameter Overview

2.2.4.2.1 PropertyTagCount

The value of the **PropertyTagCount** field is the number of elements in **PropertyTags**.

2.2.4.2.2 PropertyTags

If the **ReturnValue** is zero, the **PropertyTags** field contains a list of property tags for each property currently set on the object.

2.2.5 RopSetProperties

RopSetProperties ([MS-OXCROPS] section 2.2.8.6) updates values of properties specified in the property array field. Objects that are supported for this operation are: Message objects, Folder objects, Attachment objects and Logon objects.

2.2.5.1 Request Parameter Overview

2.2.5.1.1 PropertyValueSize

The value of the *PropertyValueSize* field is the number of bytes in *PropertyValueCount* plus the number of bytes in *PropertyValues*.

2.2.5.1.2 PropertyValueCount

The value of the *PropertyValueCount* field is the number of elements in *PropertyValues*.

2.2.5.1.3 PropertyValues

The **PropertyValues** field contains a list of **TaggedPropertyValue** elements as specified in [MS-OXCDATA] section 2.11.4.

2.2.5.2 Response Parameter Overview

2.2.5.2.1 PropertyProblemCount

The value of *PropertyProblemCount* is the number of elements in *PropertyProblems*.

2.2.5.2.2 PropertyProblems

The value of *PropertyProblems* is a list of any properties that were unable to be set, and the reason the property was unable to be set. See **PropertyProblem** in [MS-OXCDATA] section 2.7 for the format of this field.

2.2.6 RopSetPropertiesNoReplicate

RopSetPropertiesNoReplicate ([MS-OXCROPS] section 2.2.8.7) has the same parameters and works the same way as **RopSetProperties** ([MS-OXCROPS] section 2.2.8.6), with the exception that when used to set properties on a Folder Object, the set properties will not undergo folder replication. For more information on folder replication, see [MS-OXCSTOR] section 3.1.4.3. On all other objects, **RopSetPropertiesNoReplicate** works exactly the same as **RopSetProperties**.

2.2.7 RopDeleteProperties

RopDeleteProperties ([MS-OXCROPS] section 2.2.8.8) removes the values of properties specified in the *PropertyValues* field from the base object. Objects that are supported for this operation are: Message objects, Folder objects, Attachment objects and Logon objects.

2.2.7.1 Request Parameter Overview

2.2.7.1.1 PropertyTagCount

The value of **PropertyTagCount** is the number of elements in **PropertyTags**.

2.2.7.1.2 PropertyTags

PropertyTags contains a list of property tags for the properties that the client is deleting. See **PropertyTag** in [MS-OXCDATA] section 2.9.

2.2.7.2 Response Parameter Overview

2.2.7.2.1 PropertyProblemCount

The value of **PropertyProblemCount** is the number of elements in **PropertyProblems**.

2.2.7.2.2 PropertyProblems

The **PropertyProblems** field contains a list of which properties the server failed to delete, and the reason the property was unable to be deleted. See **PropertyProblem** in [MS-OXCDATA] section 2.7 for the format of this field.

2.2.8 RopDeletePropertiesNoReplicate

RopDeletePropertiesNoReplicate ([MS-OXCROPS] section 2.2.8.9) has the same parameters and works the same way as **RopDeleteProperties** ([MS-OXCROPS] section 2.2.8.8), with the exception that when used to delete properties from a Folder Object, the deleted properties will not undergo folder replication. (See [MS-OXCSTOR] section 3.1.4.3 for details on folder replication.) On all other objects, **RopDeletePropertiesNoReplicate** works exactly the same as **RopDeleteProperties**.

2.2.9 RopQueryNamedProperties

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

2.2.9.1 Request Parameter Overview

2.2.9.1.1 QueryFlags

QueryFlags is a **BYTE** bit field. Any bits not specified in the following table SHOULD be ignored by the server.

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** MUST be returned in a successful response.

2.2.9.2 Response Parameter Overview

2.2.9.2.1 IdCount

The value of *IdCount* is the number of elements in *PropertyIds* and in **PropertyNames**. *PropertyIds* MUST have *IdCount* number of elements, and **PropertyNames** MUST have *IdCount* number of elements.

2.2.9.2.2 PropertyIds

The **PropertyIds** field contains a list of Property IDs (see [MS-OXCDATA] section 2.9), one for each of the named properties which match the **PropertyGUID** requested.

2.2.9.2.3 PropertyNames

The **PropertyNames** field contains a list of Property Names in the format specified in [MS-OXCROPS] section 2.2.8. This field MUST contain *IdCount* entries. The entries in this list MUST match the order of the entries in *PropertyIds* to form Property ID to Name pairs for each named property matching the **PropertyGUID** specified in the request.

2.2.10 RopCopyProperties

RopCopyProperties ([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. This ROP is valid on objects of type Folder objects, Attachment objects and Message objects. Also, the source and destination object MUST be of the same type.

2.2.10.1 Request Parameter Overview

2.2.10.1.1 WantAsynchronous

If this field is set to zero, then the server performs the ROP synchronously. If the field is set to nonzero, then the server can either perform the ROP synchronously or asynchronously. If the server performs the ROP asynchronously, then it returns a **RopProgress** ([MS-OXCROPS] section 2.2.8.13) response instead of a **RopCopyProperties** ([MS-OXCROPS] section 2.2.8.11) response. For more details, see section 2.2.22 and section 3.1.5.

2.2.10.1.2 CopyFlags

CopyFlags is a BYTE bit field. The following table lists the possible values.<1>

Name	Value	Description
Move	0x01	If set, makes the call a move operation rather than a copy operation.
NoOverwrite	0x02	If set, any properties being set by **RopCopyProperties** ([MS-OXCROPS] section 2.2.8.11) that already have a value on the destination object will not be overwritten; otherwise, they are overwritten.

2.2.10.1.3 PropertyTagCount

The value of **PropertyTagCount** is the number of elements in **PropertyTags**.

2.2.10.1.4 PropertyTags

The **PropertyTags** field contains a list of property tags ([MS-OXCDATA] section 2.9) to be copied or moved.

2.2.10.2 Response Parameter Overview

2.2.10.2.1 PropertyProblemCount

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

2.2.10.2.2 PropertyProblems

The *PropertyProblems* field contains a list of properties that the server failed to copy and the reason each property was unable to be copied. See **PropertyProblem** in [MS-OXCDATA] section 2.7 for the format of this field. 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 in the request buffer. The *DestHandleIndex* MUST NOT be present in a response if the *ReturnValue* field is not set to **NullDestinationObject**.

2.2.11 RopCopyTo

The **RopCopyTo** ([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. The ROP is valid on Message objects, Attachment objects and Folder objects. The client can specify whether to copy the sub-objects in the source object as a flag in request buffer.

If any of the copied or moved properties already exist in the destination object, the existing properties are overwritten by the new properties, unless *CopyFlags* in the request buffer specifies the 0x02 bit. Existing information in the destination object that is not overwritten is left untouched.

2.2.11.1 Request Parameter Overview

2.2.11.1.1 WantAsynchronous

If this field is set to zero, then the server performs the ROP synchronously. If the field is set to nonzero, then the server can either perform the ROP synchronously or asynchronously. If the server performs the ROP asynchronously, then it returns a **RopProgress** ([MS-OXCROPS] section 2.2.8.13) response instead of a **RopCopyTo** ([MS-OXCROPS] section 2.2.8.12) response. For more details, see section 2.2.22 and section 3.1.5.

2.2.11.1.2 WantSubObjects

If *WantSubObjects* is nonzero then sub-objects MUST also be copied. Otherwise they are not.

2.2.11.1.3 CopyFlags

CopyFlags is a bit field of the following values.

Name	Value	Description
Move	0x01	If set, makes the call a move operation rather than a copy operation. This will delete the moved properties from the source object.
NoOverwrite	0x02	If set, any properties being set by **RopCopyTo** ([MS-OXCROPS] section 2.2.8.12) that already have a value on the destination object will not be overwritten; otherwise, they are overwritten.

2.2.11.1.4 ExcludedTagCount

The value of *ExcludedTagCount* is the number of elements in *ExcludedTags*.

2.2.11.1.5 ExcludedTags

The value of *ExcludedTags* is a list of property tags (as described in [MS-OXCDATA] section 2.9). This is a list of properties that MUST NOT be copied or moved as part of this operation.

2.2.11.2 Response Parameter Overview

2.2.11.2.1 PropertyProblemCount

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

2.2.11.2.2 PropertyProblems

The *PropertyProblems* field contains a list of which properties the server failed to copy, and the reason the property was unable to be copied. See **PropertyProblem** in [MS-OXCDATA] section 2.7 for the format of this field. 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 in the

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

2.2.12 RopGetPropertyIdsFromNames

The **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1) remote operation (ROP) is used to map abstract, client-defined property names into concrete 16-bit property IDs (15 of which are significant). A fully formed property tag is built from the 16-bit property ID and a 16-bit type code. All operations with properties (setting, getting, deleting) on messaging objects are done with the concrete 32-bit property tag. In case the predefined set of property tags does not cover a semantic need for a new client feature, the client is able to register new property IDs using this remote operation. The client provides a namespace identifier; a property identifier in either integer or string form and the server returns a property ID to use when the client wishes to make use of that property. String-form names are case-sensitive, except for names in the **PS-INTERNET-HEADERS** namespace ([MS-OXPROPS] section 1.3.2), which are always coerced to lower case.

This ROP is valid on Message objects, Attachment objects, Folder objects and Logon objects.

2.2.12.1 Request Parameter Overview

2.2.12.1.1 Flags

Name	Value	Description
Create	0x02	If set, indicates that the server MUST create new entries for any name parameters that are not found in the existing mapping set, and return existing entries for any name parameters that are found in the existing mapping set.

2.2.12.1.2 PropertyNameCount

The number of **PropertyName** structures that follow. A value of zero indicates the messaging client is querying the complete list of registered named properties.

2.2.12.1.3 PropertyNames

This field is an array of **PropertyName** ([MS-OXCDATA] section 2.6) structures, the length of which MUST be **PropertyNameCount** elements.

2.2.12.2 Response Parameter Overview

2.2.12.2.1 PropertyIdCount

The number of 16-bit property IDs that follow. The value of this parameter MUST be equal to the **PropertyNameCount** parameter unless the **PropertyNameCount** parameter was 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 property IDs. Total item length MUST be **PropertyNameCount**. For names that could not be mapped, the associated entry in the **PropertyIds** parameter MUST be zero. In this event, the **Return Code** MUST be equal to ecWarnWithErrors (0x00040380). The order of IDs in this array MUST match the order of the names specified in the **PropertyNames** parameter.

For any entries in the **PropertyNames** array with a **GUID** equal to **PS_MAPI** ([MS-OXPROPS] section 1.3.2), the resultant entry in the **PropertyIds** MUST be equal to the **LID** ([MS-OXCDATA] section 2.6.1) value from the **PropertyNames** entry. For other namespaces, the resultant entry MUST have the most significant bit set (0x8000) to indicate the property ID is a named property.

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 **Create** flag wasn't set in the **Flags** parameter.
- 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

The **RopGetNamesFromPropertyIds** ([MS-OXCROPS] section 2.2.8.2) remote operation (ROP) is used to map concrete property IDs to abstract, client-defined property names. A fully formed property tag is built from the 16-bit property ID and a 16-bit type code. All operations with properties, (setting, getting, deleting) on messaging objects are done with the concrete 32-bit property tag. In case the predefined set of property tags does not cover a semantic need for a new client feature, the client is able to register new property IDs using **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1). The client provides a namespace identifier; a property identifier in either integer or string form and the server returns a property ID to use when the client wishes to make use of that property. This remote operation is used to recover the abstract name from previously registered named property IDs.

This ROP is valid on 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.

2.2.13.1 Request Parameter Overview

2.2.13.1.1 PropertyIdCount

The number of 16-bit property IDs in *PropertyIds*.

2.2.13.1.2 PropertyIds

An array of 16-bit property IDs to map into abstract names. The length MUST be equal to **PropertyIdCount** elements. The client can pass property ID values less than 0x8000. These values will be mapped into names in the **PS-MAPI** ([MS-OXPROPS] section 1.3.2) namespace.

2.2.13.2 Response Parameter Overview

If the Return Code value is **Success** ([MS-OXCDATA] section 2.4), the fields specified in the following subsections are present.

2.2.13.2.1 PropertyNameCount

The number of elements in **PropertyNames**. Value MUST be equal to the *PropertyIdCount* parameter.

2.2.13.2.2 PropertyNames

An array of **PropertyName** ([MS-OXCDATA] section 2.6) structures.

2.2.14 RopOpenStream

RopOpenStream ([MS-OXCROPS] section 2.2.9.1) is used to open a property to perform a streaming operation. This ROP enables the client to perform various streaming operations on the specified property tag.

This ROP is valid on Folder objects, Attachment objects, and Message objects.

Only single-valued **PtypBinary**, **PtypObject**, **PtypString8**, and **PtypString** type properties are supported. Folder objects only support **PtypBinary** type properties.

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

2.2.14.1 Request Parameter Overview

2.2.14.1.1 PropertyTag

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

2.2.14.1.2 OpenModeFlags

The value of *OpenModeFlags* MUST be one of the following values.

Name	Value	Description
ReadOnly	0x00	Open stream for read-only access.
ReadWrite	0x01	Open stream for read/write access.
Create	0x02	Opens new stream, this will delete the current property value and open stream for read/write access. This is required to open a property that has not been set.
BestAccess	0x03	If the object this ROP is acting on was opened with ReadWrite access, then the stream MUST be opened with ReadWrite access. Otherwise, the stream MUST be opened with ReadOnly access.

2.2.14.2 Response Parameter Overview

2.2.14.2.1 StreamSize

StreamSize MUST be the current number of BYTES in the stream.

2.2.15 RopReadStream

RopReadStream ([MS-OXCROPS] section 2.2.9.2) is used to read the stream of bytes from a Stream object. This ROP is valid only on Stream object.

2.2.15.1 Request Parameter Overview

2.2.15.1.1 ByteCount

The value of *ByteCount* SHOULD be the size of the data buffer that will be sent back in the response. If the value is 0xBABE then the server will determine what buffer size to use in the response.

2.2.15.1.2 MaximumByteCount

This field MUST be present if and only if the value of *ByteCount* is 0xBABE. The value of *MaximumByteCount* specifies the maximum buffer size that the server can return in the response. Note that because *MaximumByteCount* can exceed the amount of data that can be returned in a single ROP response buffer, the response can span multiple ROP response buffers.

2.2.15.2 Response Parameter Overview

2.2.15.2.1 DataSize

The value of *DataSize* is the number of **BYTES** in *Data*.

If *ByteCount* was specified as 0xBABE, the value of *DataSize* is smaller than *MaximumByteCount* if the end of the stream is reached or to fit the data into the ROP response buffer.

If *ByteCount* was specified as a value other than 0xBABE, the value of *DataSize* is smaller than *ByteCount* if the end of the stream is reached or to fit the data into the ROP response buffer.

2.2.15.2.2 Data

Data contains the data read from the stream. Data MUST contain exactly *DataSize* BYTES.

2.2.16 RopWriteStream

RopWriteStream ([MS-OXCROPS] section 2.2.9.3) is used to write the stream of BYTES into a Stream object. This ROP is valid only on Stream objects. For Stream objects opened on properties on Folder objects, information written to the stream requires a **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4) before it is persisted to the Folder object. For all other Stream objects, the server will commit the stream to the parent object when the client calls **RopRelease** ([MS-OXCROPS] section 2.2.15.3) on the Stream object.

2.2.16.1 Request Parameters Overview

2.2.16.1.1 DataSize

The value of *DataSize* is the number of BYTES in *Data*.

2.2.16.1.2 Data

Data contains the data to be written to the stream.

2.2.16.2 Response Parameters Overview

2.2.16.2.1 WrittenSize

The value of *WrittenSize* is the number of BYTES actually written.

2.2.17 RopCommitStream

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

2.2.17.1 Request Parameters Overview

None.

2.2.17.2 Response Parameters Overview

None.

2.2.18 RopGetStreamSize

RopGetStreamSize ([MS-OXCROPS] section 2.2.9.5) is used to retrieve the size of the stream. This ROP is valid only on Stream objects.

2.2.18.1 Request Parameter Overview

None.

2.2.18.2 Response Parameter Overview

2.2.18.2.1 StreamSize

The value of *StreamSize* is the number of BYTES in the stream. The maximum allowed stream size is 2^31 BYTES.

2.2.19 RopSetStreamSize

The **RopSetStreamSize** ([MS-OXCROPS] section 2.2.9.6) remote operation (ROP) sets the size of a stream. This ROP is valid only on Stream objects. If the size of the stream is increased, then value of the extended stream MUST be zero. If the size of the stream is decreased, the information that extends past the end of the new size is lost.

RopSetStreamSize is not required prior to writing to the stream.

2.2.19.1 Request Parameter Overview

2.2.19.1.1 StreamSize

The value of *StreamSize* is the size that the stream will be set to. The maximum allowed stream size is 2^31 BYTES.

2.2.19.2 Response Parameter Overview

None.

2.2.20 RopSeekStream

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

2.2.20.1 Request Parameter Overview

2.2.20.1.1 Origin

The value of the *Origin* field MUST be one of the following values:

Value	Description
0x00	The new seek pointer is an offset relative to the beginning of the stream.
0x01	The new seek pointer is an offset relative to the current seek pointer location.
0x02	The new seek pointer is an offset relative to the end of the stream.

2.2.20.1.2 Offset

The value of the *Offset* field is a signed value representing the number of BYTES to offset the seek pointer from the origin. This value can be positive or negative.

2.2.20.2 Response Parameter Overview

2.2.20.2.1 NewPosition

The value of *NewPosition* is the number of BYTES from the beginning of the stream of the seek pointer.

2.2.21 RopCopyToStream

RopCopyToStream ([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 another destination stream. This ROP is valid only on Stream objects.

2.2.21.1 Request Parameter Overview

2.2.21.1.1 ByteCount

The value of *ByteCount* is the number of BYTES to copy from the source stream to the destination stream.

2.2.21.2 Response Parameter Overview

2.2.21.2.1 ReadByteCount

If **ReturnValue** is not **NullDestinationObject** ([MS-OXCDATA] section 2.4.1), the value of **ReadByteCount** MUST be the number of BYTES read from the base object. Otherwise the value is undefined.

2.2.21.2.2 WrittenByteCount

If **ReturnValue** is not **NullDestinationObject** ([MS-OXCDATA] section 2.4.1), the value of **WrittenByteCount** MUST be the number of BYTES written to the destination object. Otherwise the value 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 in the request buffer. The *DestHandleIndex* MUST NOT be present in a response if the *ReturnValue* field is not set to **NullDestinationObject**.

2.2.22 RopProgress

RopProgress ([MS-OXCROPS] section 2.2.8.13) responses report the progress status of an asynchronous operation to the client. **RopProgress** requests are used to request the status of an asynchronous operation. They also can be used to abort an in-progress operation. If the abort flag sent by the client is nonzero, then the operation is aborted.

RopProgress responses can be used as a response to the following ROPs:

- **RopCopyTo** (section 2.2.11)
- **RopCopyProperties** (section 2.2.10)
- **RopEmptyFolder** ([MS-OXCFOLD] section 2.2.1.9)
- **RopDeleteMessages** ([MS-OXCFOLD] section 2.2.1.11)
- **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)

2.2.22.1 Request Parameter Overview

2.2.22.1.1 WantCancel

WantCancel is a BYTE sized field that MUST be 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 Response Parameter Overview

2.2.22.2.1 CompletedTaskCount

CompletedTaskCount is a 4-BYTE field that SHOULD report an approximation of the number of tasks that the server has completed out of *TotalTaskCount* operations to complete the entire operation that the progress is being reported on.

2.2.22.2.2 TotalTaskCount

TotalTaskCount is a 4-BYTE field that MUST be greater than or equal to *CompletedTaskCount*. This field SHOULD represent an approximation of the number of tasks the server will complete for the operation that the progress is being reported on.

2.2.23 RopLockRegionStream

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

2.2.23.1 Request Parameter Overview

2.2.23.1.1 RegionOffset

RegionOffset is an unsigned 64-bit integer representing 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

RegionSize is an unsigned 64-bit integer representing the size of the region to be locked.

2.2.23.1.3 LockFlags

The value of the *LockFlags* field MUST be one of the following values.

Value	Description
0x001	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 Response Parameter Overview

None.

2.2.24 RopUnlockRegionStream

RopUnlockRegionStream ([MS-OXCROPS] section 2.2.9.10) is used to unlock a specified range of bytes in a Stream object. This ROP is valid only on Stream objects.

2.2.24.1 Request Parameter Overview

2.2.24.1.1 RegionOffset

RegionOffset is an unsigned 64-bit integer representing 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

RegionSize is an unsigned 64-bit integer representing the size 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** ([MS-OXCROPS] section 2.2.9.9) operation that was used to lock the region of bytes to be unlocked.

2.2.24.2 Response Parameter Overview

None.

2.2.25 RopWriteAndCommitStream

RopWriteAndCommitStream ([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 valid only on Stream objects. This ROP MUST NOT be used on Stream objects opened on properties on Folder objects.

2.2.25.1 Request Parameter Overview

2.2.25.1.1 DataSize

The value of *DataSize* is the number of BYTES in *Data*.

2.2.25.1.2 Data

Data contains the data to be written to the stream.

2.2.25.2 Response Parameter Overview

2.2.25.2.1 WrittenSize

The value of *WrittenSize* is the number of BYTES actually written.

2.2.26 RopCloneStream

RopCloneStream ([MS-OXCROPS] section 2.2.9.12) creates a new Stream object for accessing the same bytes but using a separate seek pointer. The new Stream object contains the same data as the source Stream object. Changes made to one Stream object are immediately visible in the other.

The initial setting of the seek pointer in the cloned Stream object is the same as the current setting of the seek pointer in the original Stream object at the time of the clone operation. This ROP is valid only on Stream objects.

2.2.26.1 Request Parameter Overview

None.

2.2.26.2 Response Parameter Overview

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.

3.1.1.1 Property Transactions

There are two different transaction models for property objects. Changes to properties on Folder objects and Logon objects are saved implicitly, whereas changes to Message objects are not saved until the client sends a **RopSaveChangesMessage** request ([MS-OXCROPS] section 2.2.6.3). Attachment objects are not saved until the client sends a **RopSaveChangesAttachment** request ([MS-OXCROPS] section 2.2.6.15).

3.1.1.2 Named Properties

Named properties are a set of mappings between property IDs and property names. The client can cache the property ID for any property names that it has queried using **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1), for the length of its session. Property IDs are not guaranteed to be the same for a new session. A property ID obtained for a property name is valid on any item within the Logon object.

3.1.1.3 Streams

Changes made to properties that have been opened as a stream using **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1) and written to using **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3) MUST not be persisted to the property object that property is on unless the client calls **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4) for streams opened on properties of folders, or **RopRelease** ([MS-OXCROPS] section 2.2.15.3) for streams opened on properties of attachments or messages.

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 When Client Needs to Query Data from an Object

If any of the properties the client wants are named properties, the client obtains the property IDs for those properties using **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1).

The client then uses **RopGetPropertiesSpecific** ([MS-OXCROPS] section 2.2.8.3) to query the value of those properties. The client checks the ReturnValue of the ROP, and the values of the properties returned for errors.

3.1.4.2 When Client Needs to Set Data

If any of the properties the client wants to set are named properties, the client obtains the property IDs for those properties using **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1).

The client then uses **RopSetProperties** ([MS-OXCROPS] section 2.2.8.6) to set the value of those properties. The client checks the ReturnValue of the ROP and if **PropertyProblemCount** field in the response is nonzero, then the client checks the problem array, to verify which properties failed to be set. Clients SHOULD NOT include read-only properties in the **PropertyValues** field of the ROP request buffer for **RopSetProperties**.

3.1.4.3 When the Client Needs to Query Data Larger than Will Fit in a Single ROP

The client uses a stream to read a property that came back with an error of **NotEnoughMemory** ([MS-OXCDATA] section 2.4.2) in a **PropertyProblem** ([MS-OXCDATA] section 2.7). A client can use **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1) to read a property even before attempting to read it using **RopGetPropertiesSpecific** ([MS-OXCROPS] section 2.2.8.3) if it expects the property to be big.

If the property the client wants to get is a named property, the client obtains the property ID for that property using **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1).

The client then uses **RopOpenStream** to obtain a handle to a stream on the property. The client uses ReadOnly as the value for *OpenModeFlags* if the client is just going to read from the stream. If the client wishes to write to this stream, the client MUST use BestAccess or ReadWrite. The client checks the ReturnValue of the ROP to verify that the handle was correctly retrieved.

If the client does not want to read from the start of the stream, the client MUST use **RopSeekStream** ([MS-OXCROPS] section 2.2.9.7). The client then uses one or more **RopReadStream** ([MS-OXCROPS] section 2.2.9.2) ROPs to read data from the stream. If the value of *ByteCount* is 0xBABE in the ROP request buffer, and the value of *DataSize* in the ROP response buffer is the same as the value of *MaximumByteCount* in the ROP request buffer, the client SHOULD issue another **RopReadStream** request to determine if there is further data in the stream. If the value of *ByteCount* is not 0xBABE in the ROP request buffer, and the value of *DataSize* in the ROP response buffer is the same as the value of *ByteCount* in the ROP request buffer, the client SHOULD issue another **RopReadStream** request to determine if there is further data in the stream. The client stops after it has read all the data it requires, or after **RopReadStream** returns zero as its *DataSize*.

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

3.1.4.4 When the Client Needs to Set Data Larger than Will Fit in a Single RopSetProperties

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

The client then uses **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1) to obtain a handle to the stream on the property. The client specifies ReadWrite, Create, or BestAccess as the value for

OpenModeFlags. The client checks the ReturnValue of the ROP to verify that the handle was correctly retrieved.

If the client does not wish to write from the start of the stream, the client MUST use **RopSeekStream** ([MS-OXCROPS] section 2.2.9.7). The client then uses one or more **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3) ROPs to write data to the stream.

The client uses **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4) to commit the changes to the property.

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 specified *WantAsynchronous* for any of the following ROPs the server can respond with a **RopProgress** response.

- **RopCopyTo** (section 2.2.11)
- **RopCopyProperties** (section 2.2.10)
- **RopEmptyFolder** ([MS-OXCFOLD] section 2.2.1.9)
- **RopDeleteMessages** ([MS-OXCFOLD] section 2.2.1.11)
- **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)

The server can respond with a **RopProgress** response buffer to notify the client of its current progress. When the client receives a **RopProgress** response it can use the *CompletedTaskCount* and *TotalTaskCount* 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

Provides access to property values on a property object on the server. The server MUST return the values for all properties on the object, including those set by any client, server, or computed properties. If the *WantUnicode* flag is used, the server MUST return string properties which are requested without a specified type (**PtypUnspecified**) in Unicode format.

If the *WantUnicode* flag is not used, 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 set on the Message object when it was opened if any, otherwise 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 set on the parent Message object when it was opened if any, otherwise 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.

In the case of a binary, object or **string property**, the server returns **NotEnoughMemory** ([MS-OXCDATA] section 2.4.2) for that property to indicate that the value is too large to be returned using this ROP.

The server MUST order properties in the *PropertyValues* element of the ROP response buffer in the same order in which properties are specified in the *PropertyTags* parameter. For properties larger than the requested *PropertySizeLimit*, the server MUST return a **NotEnoughMemory** error for their values in the response.

This ROP SHOULD only be supported by Message objects, Folder objects, Attachment objects and Logon objects.

3.2.5.2 Processing RopGetPropertiesAll

Provides access to all property values on a property object on the server. This ROP works the same as **RopGetPropertiesSpecific** ([MS-OXCROPS] section 2.2.8.3) except that it MUST return the values for all properties on the object.

3.2.5.3 Processing RopGetPropertiesList

The server MUST return the list of all properties currently set on the object.

This ROP SHOULD only be supported for Message objects, Folder objects, Attachment objects and Logon objects.

3.2.5.4 Processing RopSetProperties

The server MUST modify the value of the properties of the object.

For Message objects, the new value of the properties MUST be made available immediately when using the same handle. 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 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 when using 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 SHOULD only be supported for Message objects, Folder objects, Attachment objects and Logon objects.

3.2.5.5 Processing RopDeleteProperties

Removes the value of a set property. 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.

The server MUST remove the value of the property from the object.

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 SHOULD only be supported for Message objects, Folder objects, Attachment objects and Logon objects.

3.2.5.6 Processing RopQueryNamedProperties

The server MUST return the list of all named properties and their **PropertyIds**, filtered based on the parameters 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.

This ROP SHOULD only be supported for Message objects, Folder objects, and Attachment objects.

3.2.5.7 Processing RopCopyProperties

The server MUST copy the properties specified from the source object to the destination object.

If the Move flag is set in the **CopyFlags** field (section 2.2.10.1.2), the server SHOULD<2> delete the 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<3> 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. See section 3.1.5.

This ROP SHOULD only be supported for Message objects, Attachment objects, and Folder objects.

3.2.5.8 Processing RopCopyTo

This ROP works in the same way as **RopCopyProperties** ([MS-OXCROPS] section 2.2.8.11) except that it MUST copy all writable properties on the source to the destination except for the properties specified.

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 2.737) as the sub-object to be copied.

3.2.5.9 Processing RopGetNamesFromPropertyIds

This ROP SHOULD only be supported by Folder objects, Message objects, Attachment objects and Logon objects.

For each property ID in the *PropertyIds* parameter, the server MUST perform the following:

1. If the property ID value is less than 0x8000, the associated **PropertyName** ([MS-OXCDATA] section 2.6) MUST be composed:

   - A **GUID** ([MS-OXCDATA] section 2.6.1) of **PS-MAPI** ([MS-OXPROPS] section 1.3.2).
   - A **Kind** ([MS-OXCDATA] section 2.6.1) of 0x00.
   - An LID ([MS-OXCDATA] section 2.6.1) value equal to the property ID entry.

2. For property ID values that have an associated **PropertyName**, the server MUST return the **PropertyName** associated with the **PropertyId**.

3. For property ID values 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.

3.2.5.10 Processing RopGetPropertyIdsFromNames

The server MUST validate all name structures passed by the client as specified in section 2.2.12.

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.

If the **PropertyNameCount** parameter is zero, and the **RopGetPropertyIdsFromNames** ([MS-OXCROPS] section 2.2.8.1) is acting on a Logon object, the server MUST enumerate all property names associated with property IDs. Otherwise, the server MUST, for each entry in the **PropertyNames** parameter, 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* parameter has the *Create* flag bit set.
   - The user has permission to create new entries.
   - The server-imposed limit on property ID mappings specified earlier 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.

3.2.5.11 Processing RopOpenStream

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.

RopOpenStream provides access to a property in the form of a Stream object. The object returned by this ROP can then be used on subsequent ROPs, such as **RopReadStream** ([MS-OXCROPS] section 2.2.9.2), **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3) and **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4). See specific ROPs for information of which operate on streams.

This ROP SHOULD only be supported by Message objects, Attachment objects and Folder objects.

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 specified in the request. When the client calls **RopRelease** on the Stream object, and the property the Stream object is acting on is not on a Folder object, the server MUST commit the Stream object.

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

Name	Value	Description
		because Create was not specified in *OpenModeFlags*.

3.2.5.12 Processing RopReadStream

The server MUST read less or equal to the amount of data requested.

This ROP SHOULD only be supported for Stream objects.

The server MUST reply with data read from the stream reading forward from the current seek pointer and place it in the response. 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, *DataSize* SHOULD be set to zero.

3.2.5.13 Processing RopWriteStream

The server SHOULD<4> return **StreamSizeError** if it writes less than the amount requested.

This ROP SHOULD only be supported for Stream objects.

The server MUST store the data specified in the request into the 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. Calling only **RopWriteStream** ([MS-OXCROPS] section 2.2.9.3) on this stream MUST not change the value of the property specified in the **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1) request. The value of the property MUST only be changed when **RopCommitStream** ([MS-OXCROPS] section 2.2.9.4) is called on the stream, or in the case of properties on non-Folder objects, when either **RopCommitStream** or **RopRelease** ([MS-OXCROPS] section 2.2.15.3) is called, as specified in section 2.2.16.

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

This ROP SHOULD only be supported for Stream objects.

The server MUST set the property specified in the **RopOpenStream** ([MS-OXCROPS] section 2.2.9.1) request that opened this stream with the data from the stream.

3.2.5.15 Processing RopGetStreamSize

RopGetStreamSize ([MS-OXCROPS] section 2.2.9.5) MUST return the current size of the Stream object.

This ROP SHOULD only be supported by Stream objects.

3.2.5.16 Processing RopSetStreamSize

This ROP MUST set the current size of the Stream object.

This ROP SHOULD only be supported by Stream objects.

3.2.5.17 Processing RopSeekStream

This ROP SHOULD only be supported by Stream objects.

The value of the seek point associated with this Stream object MUST be modified according to the request. If the client requests the seek pointer be moved beyond 2^31 BYTES, the server MUST return **StreamSeekError**. If the client requests the seek pointer be moved beyond the end of the stream, the stream is extended, and zero filled to the new seek location.

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

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

This ROP SHOULD <5> only be supported by Stream objects.

The position of both the source and destination streams MUST be moved forward the same number of BYTES as was copied.

The server MUST read the number of BYTES requested from the source Stream object, and write those bytes into the destination Stream object.

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. Only if the *ReturnValue* is NullDestinationObject, which means that the destination object passed does not exist anymore, then the server sends the *DestinationHandleIndex* back in 4 bytes.

3.2.5.19 Processing RopProgress

If the server is not done with the asynchronous operation, then it MUST respond with a **RopProgress** response, as specified in section 2.2.22.2.<6> If *WantCancel* is 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 corresponding to the original request.

3.2.5.20 Processing RopLockRegionStream

Servers SHOULD <7> implement this ROP as follows:

If the server implements this ROP, the server MUST check for any existing locks on the requested region. 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 to 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 an **AccessDenied** error.

If a session with an expired lock calls any ROP for this Stream object that would encounter the locked region, the server MUST return a **NetworkError**.

The server MUST limit access to the locked region by other sessions as indicated by the *LockFlags* field (section 2.2.23.1.3).

3.2.5.21 Processing RopUnlockRegionStream

Servers SHOULD <8> implement this ROP as follows:

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

3.2.5.22 Processing RopWriteAndCommitStream

Servers SHOULD<9> 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.

3.2.5.23 Processing RopCloneStream

Servers SHOULD <10> implement this ROP as follows:

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

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

The following examples illustrate the byte order of ROPs in a buffer being prepared for transmission. Please note that the examples listed here only show the relevant portions of the specified ROPs; this is not the final byte sequence 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-OXCROPS] section 3. These examples assume the client has already successfully logged onto the server and opened the folder they wish to modify the **rules** on.

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 left of the colon is the offset of the following bytes into the buffer, expressed in hexadecimal notation. Following the offset is a series of up to 16 bytes, with each two character sequence describing the value of one byte in hexadecimal notation. Here, the bold byte **4d** (01001101) is located 0x83 bytes (131 bytes) from the beginning of the buffer. The dash between the eighth byte ("44") and ninth byte ("4c") has no semantic value, and serves only to distinguish the eight byte boundary for readability purposes.

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 **ULONG** with a value of 10 because the type of the **PidTagRuleSequence** property ([MS-OXPROPS] section 2.1023) is **ULONG**.

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*, 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 for the *PropertyIdCount* 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 is *PropertyIds* 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 *PropertyValues*, where each row in array consists of *PropertyTag* 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 *PropertyTags* 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 **FlaggedPropertyValues** ([MS-OXCDATA]), each consisting of a **Flag** and a **PropertyValue**. 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 contain 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.

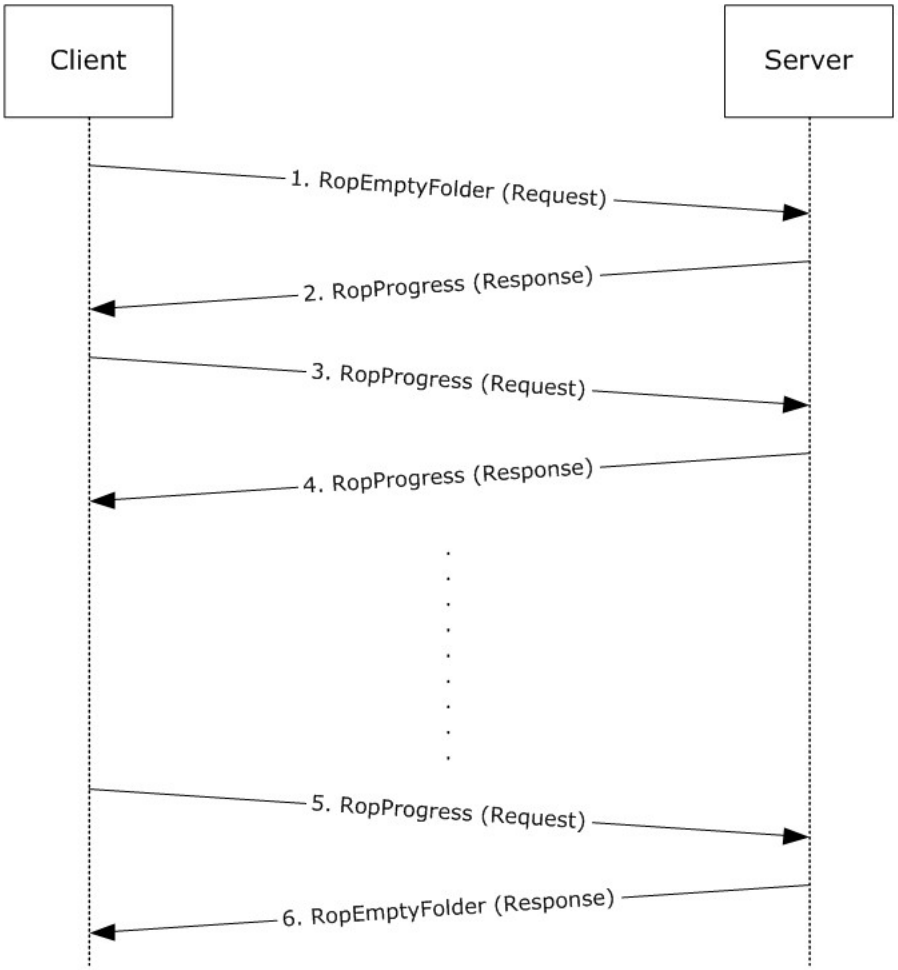

Figure 1: Sequence in which ROP are passed between client and server

1. Client sends a **RopEmptyFolder** request to the server. The 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. The ROP response buffer 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 response 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. The ROP response buffer 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 buffer 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 pertaining to the underlying RPC-based transport apply (see [MS-OXCROPS] section 5).

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 2.2.10.1.2: 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.

<2> Section 3.2.5.7: Exchange 2010 does not remove the properties from the source object.

<3> Section 3.2.5.7: Exchange 2003 and Exchange 2007 ignore invalid bits and do not return the **InvalidParameter** error.

<4> Section 3.2.5.13: Exchange 2010 returns **TooBig** if it writes less than the amount requested.

<5> Section 3.2.5.18: The initial release version of Exchange 2010 does not implement this ROP.

<6> Section 3.2.5.19: The initial release version of Exchange 2010 does not implement **RopProgress** ([MS-OXCROPS] section 2.2.8.13).

<7> Section 3.2.5.20: Exchange 2010 does not implement this ROP.

<8> Section 3.2.5.21: Exchange 2010 does not implement this ROP.

<9> Section 3.2.5.22: Exchange 2010 does not implement this ROP.

<10> 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 March 2011 and August 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
3.2.5.5 Processing RopDeleteProperties	Revised details about persisting deleted values on Attachment objects.	N	Content updated.
3.2.5.7 Processing RopCopyProperties	Added product behavior note to specify that Exchange 2010 does not remove the properties from the source object.	N	New product behavior note added.
3.2.5.7 Processing RopCopyProperties	Added product behavior note to specify that Exchange 2003 and Exchange 2007 ignore invalid bits and do not return the InvalidParameter error.	N	New product behavior note added.
3.2.5.7 Processing RopCopyProperties	Revised details about persisting changes on Attachment objects.	N	Content updated.
3.2.5.8 Processing RopCopyTo	Added the FolderCycle error to table and revised the description of the MessageCycle error.	N	Content updated.
3.2.5.10 Processing RopGetPropertyIdsFromNames	Cited [MS-OXPROPS] section 1.3.2 as the reference for property sets.	N	Content updated.
3.2.5.13 Processing RopWriteStream	Added error code TooBig to the table of error codes.	N	Content updated.
3.2.5.13 Processing RopWriteStream	Added product behavior note to specify that Exchange 2010 returns TooBig if it writes less than the amount requested.	N	New product behavior note added.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.2.5.14 Processing RopCommitStream	Deleted the STG_E_ACCESSDENIED error code and the table.	N	Content updated.
6 Appendix A: Product Behavior	Removed Exchange Server 2010 Service Pack 1 from the list of applicable products.	N	Content removed.

8 Index