[MS-OXCSTOR]: Store Object Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights**. This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.

- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.

- **Patents**. Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp/default.mspx). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@microsoft.com.

- **Trademarks**. The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

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

Preliminary Documentation. This documentation is preliminary documentation for these protocols. Since the documentation may change between this preliminary version and the final version, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Tools. This protocol documentation is intended for use in conjunction with publicly available standard specifications and networking programming art, and assumes that the reader is either familiar with the aforementioned material or has immediate access to it. A protocol specification does not require the use of Microsoft programming tools or programming environments in order for a Licensee to develop an implementation. Licensees who have access to Microsoft programming tools and environments are free to take advantage of them.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability

Table of Contents

1 Introduction

This document specifies the Store Object protocol, which is used by clients to log on to a user mailbox or public folders; read and write mailbox-level properties for that user mailbox; perform various housekeeping tasks for that mailbox; and determine availability of content for public folders.

1.1 Glossary

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

ASCII
active replica
EntryID
ESSDN
folder
LongTermID
message
messaging object
property
property tag
remote operation (ROP)
replica
replica state
REPLID
ROP request buffer
ROP response buffer
ShortTermID
special folder
store
table
Unicode

The following data types are defined in [MS-DTYP]:

Boolean
BYTE
DWORD
FILETIME
ULONG
GUID
SVREID

The following terms are specific to this document:

LogonID: An 8-bit value used to identify a logon session within a single RPC session.

interpersonal messaging subtree: The root of the hierarchy of folders commonly visible in a messaging client. This would include mailbox folders (such as Inbox and Outbox) and user-created folders, including user-created public folders.

IPM subtree: See interpersonal messaging subtree

non-interpersonal messaging subtree: The root of the hierarchy of folders not commonly visible in a messaging client. This would include server and client created folders used principally for containing operational metadata.

non-IPM subtree: See non-interpersonal messaging subtree

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

[MS-DTYP] Microsoft Corporation, "Windows Data Types", March 2007, http://go.microsoft.com/fwlink/?LinkId=111558.

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference", March 2007, http://go.microsoft.com/fwlink/?LinkId=112265.

[MS-NSPI] Microsoft Corporation, "Name Service Provider Interface (NSPI) Protocol Specification", April 2008.

[MS-OXCDATA] Microsoft Corporation, "Data Structures Protocol Specification", April 2008.

[MS-OXCFXICS] Microsoft Corporation, "Bulk Data Transfer Protocol Specification", April 2008.

[MS-OXCNOTIF] Microsoft Corporation, "Core Notifications Protocol Specification", April 2008.

[MS-OXCROPS] Microsoft Corporation, "Remote Operations (ROP) List and Encoding Protocol Specification", April 2008.

[MS-OXCRPC] Microsoft Corporation, "Wire Format Protocol Specification", April 2008.

[MS-OXDISCO] Microsoft Corporation, "Autodiscover HTTP Service Protocol Specification", April 2008.

[MS-OXGLOS] Microsoft Corporation, "Office Exchange Protocols Master Glossary", April 2008.

[MS-OXORULE] Microsoft Corporation, "E-mail Rules Protocol Specification", April 2008.

[MS-OXWOOF] Microsoft Corporation, "Out of Office (OOF) Web Service Protocol Specification", April 2008.

[MS-UCODEREF] Microsoft Corporation, "Windows Protocol Unicode Reference", July 2007, http://go.microsoft.com/fwlink/?LinkId=112413.

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

1.2.2 Informative References

None.

1.3 Protocol Overview (Synopsis)

1.3.1 Private vs. Public Stores

The client can log on to a private user mailbox for access to that single user's mailbox data (folders, messages and attachments). Once logged on, the client can perform operations on the mailbox using operations specified in this protocol. The client can also simultaneously log on to other users' mailboxes, and granted sufficient permissions by that other user, access that user's mailbox data (folders, messages and attachments) as well as perform operations on the mailbox. Additionally, the client can simultaneously log on to a public folder store.

The content within an entire private mailbox is confined to a single server. The client determines the server to log on to from global configuration data about the user. If the mailbox has been moved to another server, the client attempt to log on to the wrong server will result in an error response from that wrong server, along with a hint of which server the client should try next.

The content within the public folders store is typically spread across many different servers, and is replicated among those servers. The client determines, from the global configuration about the user, to which public folder server to log on. All the servers hosting public folders contain a complete copy of the public folders store's folder hierarchy. However, a specific server may not have the contents of any particular public folder. The set of servers which contain content for a specific folder are said to be content replicas for that folder. A client attempting to read folder content from a server which is not a content replica for that folder, will result in an error response. The client is then able to use operations specified in this

protocol to discover which servers are content replicas for the folder. After making that determination, the client would then log on to one of those servers to read or update the content for that folder.

1.3.2 Opening a Store Connection

The client must first connect to the server in question and establish an RPC context as specified in [MS-OXCRPC]. Once that connection is made, the client is then able to follow the protocol specified in this document to establish a logon session with a private mailbox, or the public folders. After the logon session is established, the client is then able, using information specified in [MS-OXCROPS], to follow the protocol specified in this document to perform various operations on the user mailbox and make discoveries about where public folder content is located. Note that establishing an RPC context and subsequently establishing a logon session are prerequisite steps for all other remote operations specified in [MS-OXCROPS].

1.4 Relationship to Other Protocols

The Store Object protocol relies on the Remote Operations (ROP) List and Encoding protocol, as specified in [MS-OXCROPS], and the Wire Format protocol, as specified in [MS-OXCRPC]. Many other Office Exchange protocols, such as the Folder Object Protocol and Message Object Protocol, both of which issue **remote operations (ROP)** to an Exchange Server, rely on this protocol in that they must first successfully complete a **RopLogon** as specified in this document.

1.5 Prerequisites/Preconditions

The Store Object Protocol specification assumes that the messaging client has previously connected to the messaging server, as specified in [MS-OXCRPC]. Except for issuing the **RopLogon** remote operation, all other remote operations to be performed assume that the messaging client has successfully completed a **RopLogon** remote operation.

1.6 Applicability Statement

One uses the Store Object Protocol when one needs to establish a connection to a specific mailbox or the public folder store. This connection is the basis for a context for other protocols that require a logon object handle.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The ROP request buffers and ROP response buffers specified by this protocol are sent to, and respectively, are received from the server using the underlying Remote Procedure Call transport specified in [MS-OXCROPS].

2.2 Message Syntax

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

2.2.1 Remote Operations

The following sections specify the fields passed in **ROP request buffers** that are specific to the Store Object Protocol. Before sending these requests to the server, the messaging client MUST be connected to the server for RopLogon as specified in [MS-OXCRPC]. For other remote operations, the messaging client MUST have successfully completed a RopLogon operation and MUST pass a valid Server object value that represents a logon object obtained from the successful completion of a RopLogon operation.

For all flags fields specified below, all bits in the bitfield not defined here MUST NOT be set by the client and MUST be ignored by the server.

2.2.1.1 RopLogon Semantics

The syntax of the RopLogon request and response is specified in [MS-OXCROPS].

The RopLogon remote operation establishes a logon session between the messaging client and the server. It is the basis of all further remote operations, and successfully completing a RopLogon is a prerequisite for all other remote operations listed in this specification to be performed.

2.2.1.1.1 Request

Several fields are passed from the client to the server, and the presence of some of them depends on flag bits present or absent in other fields. Other flags fields control server-side behavior. For any flags field, any flag values not defined here MUST NOT be set by a client and servers SHOULD reject the operation with ecRpcFormat.

The fields are:

2.2.1.1.1.1 LogonFlags

This is a flags field. Individual flag values and their meanings are specified below:

Name	Value	Description
Private	0x01	Set to logon to a private mailbox. Not set for logon to public folders.
Undercover	0x02	Value MUST be ignored by the server and is returned to the client in the response. See Response details below.
Ghosted	0x04	If the private bit is set, this bit MUST not be set by the client and MUST be ignored by the server. If this bit is unset AND the Open Flags field does not have any of the following bits set ALTERNATE_SERVER OVERRIDE_HOME_MDB IGNORE_HOME_MDB The server will look up in the Directory the default public folder database to log on to. If this server does not host that database, the ReturnValue will be ecWrongServer. See *ReturnValue* below for details. Otherwise, the server will log on to the public folder database present on the server, if there is one. If there is no public folder database on the server, *ReturnValue* will be ecLoginFailure. See *ReturnValue* below for details. For successful returns, this bit is returned unchanged in the response. See Response details below.
SplProcess	0x08	Value MUST be ignored by the server and is returned to the client in the response. See Response details below.

2.2.1.1.1.2 OpenFlags

This is a flags field. Individual flag values and their meanings are specified below:

Name	Value	Description
PUBLIC	0x00000002	If set, open the public folders store, Otherwise open a user mailbox.
HOME_LOGON	0x00000004	Opens the home logon of the user's

Name	Value	Description
		information store. Should be used when a separate logon to the user's information store is required. For example, if your application tracks the read status of messages in a public folder on a per-user basis, it must log on to the user's mailbox in addition to the public folder to mark certain items as read.
TAKE_OWNERSHIP	0x00000008	Opens the information store as the owner of the store, by using the identity of the owner of the store.
ALTERNATE_SERVER	0x00000100	Requests a private server to give an alternate public server.
IGNORE_HOME_MDB	0x00000200	This is used only for public logon. Normally, the client is only allowed to log on to the user's configured default public folder server. Attempts to log on to any other server will result in a redirect back to the default. If the client needs to log on to some other public folder server, the client MUST set this bit to indicate to the server that the default override is requested.
NO_MAIL	0x00000400	Requests a non-messaging Logon session. Non-messaging sessions allow clients to access the store, but do not allow messages to be sent or received.
USE_PER_MDB_REPLID MAPPING	0x01000000	Messaging clients SHOULD set this bit when logging on to a private mailbox. For logons to a public folder store, the server MUST ignore the value of this bit and always assume this bit is set. See section 2.2.1.1.1.2.1 for a detailed specification of how this bit controls

Name	Value	Description
		server behavior.

2.2.1.1.1.2.1 USE_PER_MDB_REPLID_MAPPING Details

Clients that set this bit in the *OpenFlags* field are communicating to the server that any client-side caching of REPLID to REPLGUID mappings and/or caching of Named Property mappings are maintained per logon session, and not per server connection. Even if the messaging client does not intend to keep any client-side caches, it MUST set this bit. If a messaging client issues a **RopLogon** with this bit unset, the server MUST assume all REPLIDs coming from the client, in any context that uses short-term IDs, were originated from the first mailbox database referenced on the current connection (known as the "reference database"). The same holds true for any named properties used in any context that uses property IDs. The server MUST perform the following actions in all contexts:

- Upon commencing each remote operation that pass item IDs or change numbers, all item IDs and change numbers MUST have their REPLID values converted from the logon's reference database to the database that will actually hold the item being identified. This is accomplished by first invoking the logic of RopLongTermIdFromId, using the logon's reference database's REPLID and REPLGUID to-and-from mapping table, and then invoking the logic of RopIdFromLongTermId using the logon's actual database's REPLID and REPLGUID to-and-from mapping table.
- Upon concluding each remote operation, the reverse MUST be performed – all item IDs and change numbers being returned to the client must be converted using the above described process, first using the actual database to map the REPLIDs to REPLGUIDs, and then the reference database to map those REPLGUIDs back to REPLIDs.
- Similarly, upon commencing each remote operation that pass property IDs, either directly, or embedded in other structures (such as restrictions), the server MUST first identify any named properties that are included, and then invoke the logic of RopGetNamesFromPropertyIds using the reference database's named property table, subsequently invoking RopGetPropertyIdsFromNames on the actual database's named property table.
- Finally, upon completing each remote operation that returns property IDs, either directly or embedded in other structures (such as restrictions), the server MUST first identify any named properties that are included, and then invoke the logic of RopGetNamesFromPropertyIds using the actual database's named property table, subsequently invoking RopGetPropertyIdsFromNames on the reference database's named property table.

2.2.1.1.1.3 StoreState

Unused. SHOULD be set to zero.

2.2.1.1.1.4 EssdnSize

MUST be the length in bytes of the *Essdn* string, including its zero-byte terminator. Clients MUST pass zero for public folders logons.

2.2.1.1.1.5 Essdn

An ASCII string uniquely identifying a mailbox to log on to within some global configuration. The mailbox descriptor in the global configuration will contain enough other data to identify the correct server hosting the user's mailbox, as well as how to find that specific mailbox on that server. String MUST include the zero terminating byte. String length (including zero terminator) MUST be equal to *EssdnSize*. See [MS-NSPI] and [MS-OXDISCO] on how to obtain this string for any specific user. Any user object obtained using either protocol specified in [MS-NSPI] or [MS-OXDISCO] will have an attribute on it named "legacyExchangeDN". The string data in this attribute is the string to use in this field.

2.2.1.1.2 Redirect Response

These return values are sent when the *ReturnCode* is ecWrongServer.

2.2.1.1.2.1 LogonFlags

Value MUST be composed of the values of *Private*, *Undercover*, and *Ghosted* bits passed to the server in the *LogonFlags*field.

2.2.1.1.2.2 EssdnSize

The length of the *Essdn* string that follows, including its zero terminator byte.

2.2.1.1.2.3 Essdn

The ESSDN of the correct server the client should be connecting to, as this server either no longer hosts the requested mailbox (it was moved), or was the wrong server to connect to for access to public folders. String MUST include the zero terminating byte. String length (including zero terminator) MUST be equal to *Size*.

2.2.1.1.3 Success Response for Private Mailbox

These return values are sent only when the *Private* bit is set in the *LogonFlags* field.

2.2.1.1.3.1 LogonFlags

Value MUST be composed of the values of *Private*, *Undercover*, and *Ghosted* bits passed to the server in the *LogonFlags* field.

2.2.1.1.3.2 Private Interpersonal Messaging Folder IDs

Mailbox Root Folder. All other folders listed here are direct or indirect children of this folder.

Deferred Action Folder

Spooler Queue

Interpersonal Messages Subtree (root of the user-visible portion of the folder hierarchy)

Inbox

Outbox

Sent Items

Deleted Items

Common Views

Schedule

Search

Views

Shortcuts

2.2.1.1.3.3 ResponseFlags

This is a flags field. Individual flag values and their meanings are specified below:

Name	Value	Description
Reserved	0x01	MUST be set.
OwnerRight	0x02	If set, the user has Full Owner or View Admin rights over the mailbox.
SendAsRight	0x04	If set, the user has the right to send mail from this mailbox..
OOF	0x10	Indicates if Out Of Office is set for the mailbox. For more information about the Out Of Office state, see [MS-OXWOOF].

2.2.1.1.3.4 MailboxGuid

The GUID of the mailbox that was logged on to.

2.2.1.1.3.5 ReplId

Short form of the *ReplGuid*.

2.2.1.1.3.6 ReplGuid

GUID used to identify the source of the REPLID to GUID mapping and named property mappings. This value is randomly assigned to a database when it is created and is an integral part of all IDs assigned in the database. It is used for forming **LongTermIDs**. This value is derived from the reference database. See section 2.2.1.1.1.2.1 for details of the reference database. If a client chooses to cache mapping information, it MUST cache this GUID with the mapping information and only use the mapping information which it had previously cached if the cached GUID matches this GUID.

2.2.1.1.3.7 LogonTime

The UTC time on the server when the logon was performed. The format of this field is specified in [MS-OXCROPS].

2.2.1.1.3.8 GWARTTime

The UTC time on the server when the list of supported address types was last updated. For more information on address types, see [MS-OXCMSG].

2.2.1.1.3.9 StoreState

If the mailbox currently has any active search folders present within, this bitfield MUST have the *STORE_HAS_SEARCHES* flag set. All other bits MUST NOT be set by the server and SHOULD be ignored by the client.

Name	Value	Description
STORE_HAS_SEARCHES	0x010000000	Indicates whether the mailbox has active search folders being populated. For more information on search folders, see [MS-OXCFOLD].

2.2.1.1.4 Success Response for Public Folders

These return values are sent only when the *Private* bit is not set in the *LogonFlags* field.

2.2.1.1.4.1 LogonFlags

Value MUST be composed of the values of *Private*, *Undercover*, and *Ghosted* bits passed to the server in the *LogonFlags* field.

2.2.1.1.4.2 Public Folder IDs

Public Folders Root Folder. All other folders listed here are direct or indirect children of this folder.

Interpersonal Messages Subtree

Non-Interpersonal Messages Subtree (so-called “system” folders)

EForms Registry Root Folder

Free/Busy Data Root Folder

Offline Address Book Data Root Folder

EForms Registry for the user’s locale

Local Site’s Free/Busy Data Folder

Local Site’s Offline Address Book Data Folder

NNTP Article Index Folder

Empty

Empty

Empty

2.2.1.1.4.3 ReplId

Short form of the *ReplGuid*.

2.2.1.1.4.4 ReplGuid

GUID used to identify the origin of ID and named property mappings. This value is randomly assigned to a database when it is created and is an integral part of all IDs assigned in the database. It is used for forming LongTermIDs.

2.2.1.1.4.5 PerUserGuid

Used by the client to track if the cached per-user read/unread information has been changed. The client can compare this value to any previous value from an older logon to determine that any locally cached info is stale and should be discarded.

2.2.1.1.5 Return Codes

The following table describes the common return codes that are returned for **RopLogon**. Other return codes are possible, and are specified in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecLoginFailure	0x80040111	Possible cases:

Name	Value	Meaning
		Private logon is requested without a mailbox DN.
ecUnknownUser	0x000003EB	The user specified by *UserEssdn* is unknown to the system.
ecUnknownCodePage	0x000003EF	The code page for this session is unknown.
ecMailboxDisabled	0x0000096C	The user account is marked as disabled.
ecMailboxInTransit	0x0000050C	Mailbox is in transit; logon is not allowed
ecWrongServer	0x00000478	The requested MDB for logon is not the user's home MDB. If the user wants to logon in any other MDB than his home MDB, then the Open flag **OVERRIDE_HOME_MDB**needs be set.
ecProfileNotConfigured	0x0000011C	A normal user tries to log on in a non-home MDB and the Logon flag is not set with Open flag **USE_ADMIN_PRIVILEGE**.
ecAccessDenied	0x80070005	User doesn't have sufficient permissions to the mailbox.
ecLoginPerm	0x000003F2	A user without owner permission for an uncreated mailbox attempts to create the mailbox.
ecServerPaused	0x0000047F	The client has attempted too many logon attempts, each of which resulted in an ecWrongServer response.

2.2.1.2 RopGetReceiveFolder Semantics

The syntax of the RopGetReceiveFolder request and response is specified in [MS-OXCROPS].

The RopGetReceiveFolder remote operation (ROP) is used to determine the delivery folder for messages of a specific message class when they are delivered to a mailbox. The operation takes as a field the message class string to test and returns the folder ID where messages of that class and all subclasses will be delivered. It also returns the specific parent message class configured to deliver to that folder.

2.2.1.2.1 Request

This operation is only valid when the logon object refers to a private mailbox logon.

2.2.1.2.1.1 MessageClass

The message class string to test. String MUST include the zero terminator byte. Testing MUST be done case-insensitive. String MUST meet the following requirements:

- MUST use ASCII encoding
- Length (including the zero byte terminator) MUST be greater than zero and less than or equal to 255
- Each character value in the string must be in the numeric range of 32 to 126, inclusive.
- MUST NOT begin with a period (“.”)
- MUST NOT end with a period
- MUST NOT contain adjacent periods.

2.2.1.2.2 Response

2.2.1.2.2.1 FolderID

The folder ID of the folder where messages of class *MessageClass* will be delivered.MUST be a folder within the user’s mailbox.

2.2.1.2.2.2 ExplicitMessageClass

The message class string actually configured for delivery to the folder. String includes the zero terminator byte. Value must meet the following requirements:

- MUST use ASCII encoding
- Length (including the zero byte terminator) MUST be greater than zero and less than or equal to 255
- Each character value in the string must be in the numeric range of 32 to 126, inclusive.
- MUST NOT begin with a period (“.”)
- MUST NOT end with a period
- MUST NOT contain adjacent periods

Server can return the *MessageClass* string as originally configured by the client, coerced into all upper case, or all lower case.

For example, if the client sends a *MessageClass* of "IPM.Schedule.Meeting.Request", *ExplicitMessageClass*may be returned as "IPM.Schedule.Meeting", which implies that *all* messages that share the prefix string (ie, are a subclass of) "IPM.Schedule.Meeting" will be delivered to this folder. In this same example, if a client sends a *MessageClass* of "IPM.Schedule.Meeting", *MessageClass*will be returned as "IPM.Schedule.Meeting"

The server MUST return the actual configured message class which is the longest prefix string of the *MessageClass*field.

2.2.1.2.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecInvalidParam	0x80070057	*MessageClass* does not conform to the format requirements specified above
ecNoReceiveFolder	0x00000463	There is no configured receive folder that exactly matches *MessageClass*, nor is a prefix string of *MessageClass*
ecNotSupported	0x80040102	The remote operation was not performed against a private mailbox logon.

2.2.1.3 RopSetReceiveFolder Semantics

The syntax of the RopSetReceiveFolder request and response is specified in [MS-OXCROPS].

The RopSetReceiveFolder remote operation (ROP) is used to establish the delivery folder for messages that have a message class string which itself has a prefix of a given string. The operation takes as a field the message class string to set and the delivery folder ID for messages which have that string as a prefix string match to the message's message class string.

Multiple message classes are permitted to deliver to the same folder. A messaging client can change an existing receive folder configuration for a message class by simply issuing this ROP with a different *FolderID* value.

The server MUST record, in the Universal Coordinated Time timezone, the time the entry was created or modified so that it can be retrieved from the ROP RopGetReceiveFolderTable.

2.2.1.3.1 Request

This operation MUST be issued against a private mailbox logon.

2.2.1.3.1.1 FolderID

The Folder ID of the desired delivery folder for the *MessageClass* class and all non-specifically configured subclasses of that class. A value of all zeros means the server MUST remove any previously configured entry for the given message class.

2.2.1.3.1.2 MessageClass

The string identifying the message class to set the delivery folder for. The string MUST include the zero byte terminator. The string MUST comply with all of the following restrictions:

- MUST use ASCII encoding
- Length (including the zero byte terminator) MUST be greater than zero and less than or equal to 255
- Each character value in the string must be in the numeric range of 32 to 126, inclusive.
- MUST NOT begin with a period (".")
- MUST NOT end with a period
- MUST NOT contain adjacent periods

The *MessageClass* string is compared, case-insensitive, to all existing configured entries. Prefix string compares are not done. If an existing entry matches, the new *FolderID* value replaces the currently configured value. Otherwise, a new entry is added.

2.2.1.3.2 Response

There are no additional fields other than the return code for this operation.

2.2.1.3.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecAccessDenied	0x80070005	Client has attempted to change the receive folder for the "IPM" or "Report.IPM" classes.
ecInvalidParam	0x80070057	The *MessageClass* string does not conform to the requirements outlined above
ecError	0x80004005	*FolderID* is all zeros AND *MessageClass* is zero length
ecNotSupported	0x80040102	The remote operation was not performed against a private mailbox logon.

2.2.1.4 RopGetReceiveFolderTable Semantics

The syntax of the RopGetReceiveFolderTable request and response is specified in [MS-OXCROPS].

The RopGetReceiveFolderTable remote operation (ROP) is used to obtain a comprehensive list of all configured MessageClass to delivery folder entries. The return data consists of logical "rows" of data, each row consisting of three "columns" of values. There is one row for each configured entry, and within each row, there appears the Message Class, FolderID and LastModificationTime for the entry.

2.2.1.4.1 Request

There are no explicit fields for this operation. The data to retrieve is limited to the mailbox linked to the Logon passed as part of the normal ROP request process. This operation MUST be issued against a private mailbox logon.

2.2.1.4.2 Response

2.2.1.4.2.1 RowCount

The number of rows in the table. The rows themselves can be returned in any order.

2.2.1.4.2.2 Rows

An array of rows in the table. The format of each row is a PropertyRow, specified in[MS-OXCDATA].

The properties present in each row MUST be returned in the following order and MUST contain only and all of the following:

2.2.1.4.2.2.1 FolderId

The folder ID of the folder where messages of class *MessageClass* will be delivered.MUST be a folder within the user's mailbox.

2.2.1.4.2.2.2 MessageClass

The message class string configured for delivery to the folder. String includes the zero terminator byte. Value must meet the following requirements:

- MUST use ASCII encoding
- Length (including the zero byte terminator) MUST be greater than zero and less than or equal to 255
- Each character value in the string must be in the numeric range of 32 to 126, inclusive.
- MUST NOT begin with a period (".")
- MUST NOT end with a period
- MUST NOT contain adjacent periods

Server MAY return the *MessageClass* string as originally configured by the client, coerced into all upper case, or coerced into all lower case. The string MUST be ASCII.

2.2.1.4.2.2.3 LastModification

The UTC time of when the entry was created or last modified.

2.2.1.4.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecNoReceiveFolder	0x00000463	There are no configured receive folder entries.

Name	Value	Meaning
ecNotSupported	0x80040102	The remote operation was not performed against a private mailbox logon.

2.2.1.5 RopGetStoreState Semantics

The syntax of the RopGetStoreState request and response is specified in [MS-OXCROPS].

The RopGetStoreState remote operation (ROP) is used to obtain state information about the current mailbox.

2.2.1.5.1 Request

There are no explicit fields for this operation. The data to retrieve is limited to the mailbox linked to the logon passed as part of the normal ROP request process. This operation MUST be issued against a private mailbox logon.

2.2.1.5.2 Response

2.2.1.5.2.1 StoreState

If the mailbox currently has any active search folders present within, this bitfield MUST have the *STORE_HAS_SEARCHES* flag set. All other bits MUST NOT be set.

2.2.1.5.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecNotSupported	0x80040102	The remote operation was not performed against a private mailbox logon.

2.2.1.6 RopGetOwningServers Semantics

The syntax of the RopGetOwningServers request and response is specified in [MS-OXCROPS].

The RopGetOwningServers remote operation (ROP) is used to obtain the set of servers which host content for a replicated public folder.

When attempting to read content from a public folder on a specific server, the operation may fail with ecNoReplicaHere (0x00000468). This happens if the server being asked does not contain a replica copy of the data. In that event, the messaging client issues this ROP to obtain the set of servers which do in fact contain the data. The returned data is an ordered set of server names, sorted by the configured network costs which connect this server to each of those. The cost data can come from a specific configuration for this server, or can come inferred from other network configuration settings (specific site router costs, for example).

2.2.1.6.1 Request

This operation SHOULD be issued against a public folders logon.

2.2.1.6.1.1 FolderID

The FID of the public folder for which to obtain the replica set server names.

2.2.1.6.2 Response

2.2.1.6.2.1 OwningServersCount

The number of strings which follow.

2.2.1.6.2.2 CheapServersCount

The number of entries at the front of the list which have the same lowest network cost. MUST be less than or equal to *OwningServersCount* and MUST be greater than zero if *OwningServersCount* is greater than zero.

2.2.1.6.2.3 OwningServers

An array of ASCII strings. Each entry MUST include the zero terminator byte. The number of strings MUST be equal to *OwningServersCount*. The entries are sorted by the server's notion of the network cost to connect to each of the servers in the list. The source of these network costs MAY be whatever configuration source the server finds most appropriate.

Each string is the ESSDN of a public folder database which itself hosts an **active replica** of the content of the folder. Folders can exist in one of several replica states (Active, Pending Removal, and Inactive, among others) and only those in the Active state are returned.

The server MAY remove active replicas from the list if it deems them "too expensive" for the client to attempt a connection, or if other configuration settings have identified that server as unavailable to clients for some reason. "Too expensive" is defined by the implementation.

2.2.1.6.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecNoReplicaAvailable	0x00000469	There are no active replicas for the folder OR the only available replicas have been deemed "too expensive" to reach or are otherwise deemed "unavailable".
ecNotFound	0x8004010F	The *FolderID* could not be found in the public folder database.

2.2.1.7 RopPublicFolderIsGhosted Semantics

The syntax of the RopPublicFolderIsGhosted request and response is specified in [MS-OXCROPS].

The RopPublicFolderIsGhosted remote operation (ROP) is used to obtain the replication state for a folder on the current server. Folders can exist in one of several replica states (Active, Pending Removal, and Inactive, among others).The remote operation returns TRUE if the server is *not* an active replica for the folder and FALSE if the server *is* an active replica.

2.2.1.7.1 Request

This operation SHOULD only be issued against a public folders logon. Clients MAY issue it against a private mailbox logon. The server MUST always return FALSE for *GhostedState* if the client issues this operation against a private mailbox logon (for any folder), or the IPM or non-IPM subtree root folders of the public store.

2.2.1.7.1.1 FolderID

The ID of a public folder for which to obtain the ghosted state.

2.2.1.7.2 Response

2.2.1.7.2.1 IsGhosted

A Boolean property. TRUE if the server is not an active replica of the folder. Further return values are included only if *IsGhosted* is true.

2.2.1.7.2.2 OwningServersCount

The number of strings which follow.

2.2.1.7.2.3 CheapServersCount

The number of entries at the front of the list which have the same lowest network cost. MUST be less than or equal to *OwningServersCount* and MUST be greater than zero if *OwningServersCount* is greater than zero.

2.2.1.7.2.4 Servers

An array of ASCII strings. Each entry MUST include the zero terminator byte. The number of strings MUST be equal to *OwningServersCount*. The entries are sorted by the server's notion of the network cost to connect to each of the servers in the list. The source of these network costs MAY be whatever configuration source the server finds most appropriate.

Each string is the ESSDN of a public folder database which itself hosts an active replica of the content of the folder. Folders can exist in one of several replica states (Active, Pending Removal, and Inactive, among others) and only those in the Active state are returned.

The server MAY remove active replicas from the list if it deems them "too expensive" for the client to attempt a connection, or if other configuration settings have identified that server as unavailable to clients for some reason. "Too expensive" is defined by the implementation.

2.2.1.7.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecNoReplicaAvailable	0x00000469	There are no active replicas for the folder OR the only available replicas have been deemed "too expensive" to reach or are otherwise deemed "unavailable". This error can only occur if the server itself is not an active replica.
ecNotFound	0x8004010F	The *FolderID* could not be found in the public folder database.

2.2.1.8 RopLongTermIdFromId Semantics

The syntax of the RopLongTermIdFromIdrequest and response is specified in [MS-OXCROPS].

The RopLongTermIdFromId remote operation (ROP) is used to obtain the “long-term”version of an ID, given the “short-term” version of that ID. For details, see [MS-OXCDATA].

2.2.1.8.1 Request

2.2.1.8.1.1 ObjectId

The ID to map to long-term. The 16-bit REPLID portion of the ID MUST be a valid entry in the REPLID and REPLGUID to-and-from mapping table.

2.2.1.8.2 Response

2.2.1.8.2.1 LongTermId

The same ID, with the REPLID mapped to the associated REPLGUID. The server MUST map the same REPLID to the same REPLGUID every time it is queried. Other servers MAY map a particular REPLID to a different REPLGUID than this server would do, but they too MUST map any particular REPLID to the same REPLGUID value every time they are queried.

2.2.1.8.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecNotFound	0x8004010F	The REPLID portion of *ID* could not be found in the REPLID and REPLGUID to-and-from mapping table.

2.2.1.9 RopIdFromLongTermId Semantics

The syntax of the RopIdFromLongTermId request and response is specified in [MS-OXCROPS].

The RopIdFromLongTermId remote operation (ROP) is used to obtain the “short-term” version of an ID, given the “long-term” version of that ID.

2.2.1.9.1 Request

2.2.1.9.1.1 LongTermId

The ID to map to short-term form. If the REPLGUID portion of the ID is already present in the REPLID and REPLGUID to-and-from mapping table, the associated REPLID is used to form the return value. If the REPLGUID is not present in the mapping table, a new entry is added, and the newly assigned REPLID is used to form the return value.

2.2.1.9.2 Response

2.2.1.9.2.1 ObjectId

The same ID, with the REPLGUID mapped to the associated REPLID. The server MUST map the same REPLGUID to the same REPLID every time it is queried. Other servers MAY map a particular REPLGUID to a different REPLID than this server would do, but they too MUST map any particular REPLGUID to the same REPLID value every time they are queried.

2.2.1.9.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.

2.2.1.10 RopGetPerUserLongTermIds Semantics

The syntax of the RopGetPerUserLongTermIds request and response is specified in the [MS-OXCROPS].

The RopGetPerUserLongTermIds remote operation (ROP) is used to obtain the LongTermIDs of folders in a public folders store which contain per-user read/unread data identified by a REPLGUID.

2.2.1.10.1 Request

This ROP MUST be issued against a logon that was made to a private mailbox.

2.2.1.10.1.1 ReplGuid

Identifies which replica database for which the client is querying data. This GUID is obtained from the result of a RopLogon issued against a public store. See RopLogon above, *ReplGuid* return value.

2.2.1.10.2 Response

2.2.1.10.2.1 LongTermIdCount

The count of entries in the following array. MAY be zero.

2.2.1.10.2.2 LongTermIds

An array of LongTermIDs of folders in the public store for which this user has cached read/unread information. The number of items in this array MUST be the value of *LongTermIdCount*.

2.2.1.10.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecNotSupported	0x80040102	Remote operation was attempted against a public folders logon.

2.2.1.11 RopGetPerUserGuid Semantics

The syntax of the RopGetPerUserGuid request and response is specified in [MS-OXCROPS].

The RopGetPerUserGuid remote operation (ROP) is used to obtain the REPLGUID of cached per-user read/unread data for a specific public folder. The returned GUID value allows the client to correlate the cached data with whatever replica server the client is currently communicating with. Typically, if the cached GUID value does not match the current REPLGUID the client is logged on to for public folder access, it means that the client has been referred to a server different from the one it last cached the data from. The client would then issue a RopWritePerUserInformation with the locally cached data so that per-user read/unread info on the new replica will now match what the user last saw when connected to the old replica.

2.2.1.11.1 Request

This ROP MUST be issued against a logon that was made to a private mailbox.

2.2.1.11.1.1 LongTermId

The LongTermIDof the folder to query.

2.2.1.11.2 Response

2.2.1.11.2.1 ReplGuid

The REPLGUID of the last database for which relevant read/unread information was obtained. This GUID is obtained from the result of a RopLogon issued against a public store. See RopLogon above, *ReplGuid* return value.

2.2.1.11.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecNotSupported	0x80040102	Remote operation was attempted against a public folders logon.
ecNotFound	0x8004010F	The public folder identified by *LongTermId* could not be found

2.2.1.12 RopReadPerUserInformation Semantics

The syntax of the RopReadPerUserInformation request and response is specified in [MS-OXCROPS].

The RopReadPerUserInformation remote operation (ROP) is used to obtain the set of change numbers of messages the user has read in a specific folder. Note, this is not a set of message IDs, but rather the change numbers of the messages at the time they were read. Messages that are modified receive a new change number and hence fall out of the set of read messages. The user will see these modified messages marked as unread.

The format of a serialized change number set is identical to the format of a serialized IDSET with REPLGUID and is specified in [MS-OXCFXICS].

The size of the return data potentially exceeds the maximum amount of data that can be communicated in a single remote operation. For this reason, the operation is designed to stream the data to the client by having the client invoke this remote operation multiple times. Because the server can cache interim data across client calls, the client MUST complete the entire streaming operation for the data of one folder before commencing streaming operations for another folder on the same server logon. The server cannot distinguish between a client choosing to abort reading data from one folder before commencing reading from another versus doing this by accident. In the event the client does not properly prevent simultaneous

access, the server MAY return data to the client that's potentially confusing and that could lead to corrupted data.

When this remote operation is issued against a private mailbox logon, cached info for the folder is retrieved. When issued against a public folders logon, the current read/unread information is retrieved.

Used in conjunction with RopWritePerUserInformation, the client is able to move read/unread information from one public folder replica to another. For example, the client could, periodically or on a specific user action, query the public logon for read/unread information for a specific public folder by issuing RopReadPerUserInformation. It would then issue RopWritePerUserInformation against the private mailbox logon, sending the same data back to the server. This effectively saves the read/unread data in the user's mailbox. Later, when the user re-visits the public folder, the client would issue RopReadPerUserInformation against the private mailbox logon to retrieve the cached info for the folder and then issue RopWritePerUserInformation to the public folders logon to save back to the public database. In the event the client was referred to another server than had been the first time, this sequence of operations allows the user to see the same set of unread messages as the last time they had visited the folder.

2.2.1.12.1 Request

2.2.1.12.1.1 FolderId

The LongTermIDof the folder to query.

2.2.1.12.1.2 WantIfChanged

Indicator to the server to send return data only if it has changed since the last successful download.

2.2.1.12.1.3 DataOffset

The offset into the stream of data the client wants to begin reading. Value MUST be greater than or equal to zero and MUST be less than the total size of the data.

2.2.1.12.1.4 MaxDataSize

The maximum amount of data to return to the client. The server MAY return less than requested. The client MAY specify a value of zero, which indicates the server should use a suitable default value.

2.2.1.12.2 Response

2.2.1.12.2.1 HasFinished

Indicator that this is the last block of data to be returned. The client SHOULD NOT issue another RopReadPerUserInformation for the same folder. MUST be true if *WantIfChanged* is true and the underlying data has not changed.

2.2.1.12.2.2 DataSize

The size in bytes of the data being returned. MUST be less than or equal to *MaxDataSize*. MUST be zero if *WantIfChanged* is true and the underlying data has not changed.

2.2.1.12.2.3 Data

The actual data being returned. Size MUST be *DataSize*.

2.2.1.12.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecRpcFormat	0x000004B6	*DataOffset* was less than zero.
ecError	0x80004005	*DataOffset* was greater than the data size
ecNotFound	0x8004010F	The public folder identified by *FolderId* could not be found.

2.2.1.13 RopWritePerUserInformation Semantics

The syntax of the RopWritePerUserInformation request and response is specified in [MS-OXCROPS].

The RopWritePerUserInformation remote operation (ROP) is used to establish the set of change numbers of messages the user has read in a specific folder. Note, this is not a set of message IDs, but rather the change numbers of the messages at the time they were read. Messages that are modified receive a new change number, and hence, fall out of the set of read messages. The user will see these modified messages marked as unread.

The format of a serialized change number set is identical to the format of a serialized IDSET with REPLGUID and is specified in [MS-OXCFXICS].

The size of the data potentially exceeds the maximum amount of data that can be communicated in a single remote operation. For this reason, the operation is designed to stream the data to the server by having the client invoke this remote operation multiple times.

Because the server MAY cache interim data across client calls, the client MUST complete the entire streaming operation for the data of one folder before commencing streaming operations for another folder on the same server logon. The server MUST dispose of partial data if it detects the client has changed target folders before indicating to the server that the first folder's data is complete.

When this remote operation is issued against a private mailbox logon, cached info for the folder is saved. When issued against a public folders logon, the current read/unread information is established.

Used in conjunction with RopReadPerUserInformation, the client is able to move read/unread information from one public folder replica to another. For example, the client could, periodically or on a specific user action, query the public logon for read/unread information for a specific public folder by issuing RopReadPerUserInformation. It would then issue RopWritePerUserInformation against the private mailbox logon, sending the same data to the mailbox server. This effectively saves the read/unread data in the user's mailbox. Later, when the user re-visits the public folder, the client would issue RopReadPerUserInformation against the private mailbox logon to retrieve the cached info for the folder and then issue RopWritePerUserInformation to the public folders logon to save back onto the public database. In the event the client was referred to another server than had been the first time, this sequence of operations allows the user to see the same set of unread messages as the last time they had visited the folder.

2.2.1.13.1 Request

2.2.1.13.1.1 FolderId

The LongTermID of the folder for which data is being saved.

2.2.1.13.1.2 HasFinished

Indicator that this is the last block of data to be written. When the client issues this remote operation with this value set to true, the server SHOULD validate the data before committing it to storage.

2.2.1.13.1.3 DataOffset

The offset into the stream where this block of data is to be written. MUST be equal to the total size of the data previously written.

2.2.1.13.1.4 DataSize

The size in byte of the data being written.

2.2.1.13.1.5 Data

The data to write. Size MUST be equal to *DataSize*.

2.2.1.13.1.6 ReplGuid

MUST NOT be present for operations against public folders logons. MUST be present when *WriteOffset* is zero. MUST NOT be present when *WriteOffset* is not zero. Identifies which public database was the source of this data. The value is the REPLGUID of the last database for which relevant read/unread information was obtained. This GUID is obtained from the result of a RopLogon issued against a public store. See RopLogon above, *ReplGuid* return value.

2.2.1.13.2 Response

There are no return values for this operation.

2.2.1.13.3 Return Codes

All remote operations have an error return code. Upon error-free return, this return code MUST be zero. Further output values then follow. For error returns, the value will most commonly come from the Error Codes table below. Other possible error codes are listed in [MS-OXCDATA].

Name	Value	Meaning
ecNone	0x00000000	Success.
ecError	0x80004005	*DataOffset* didn't match the size of the data written so far OR *FolderId* didn't match the value on the previous call, AND THEN *DataOffset* wasn't zero.
ecFmtError	0x000004ED	The data cumulatively written could not be parsed as a proper serialized IDSET with REPLGUID, as specified in [MS-OXCFXICS].
ecNotFound	0x8004010F	The public folder identified by *FolderId* could not be found.

2.2.2 Logon-Specific Properties

The following properties are available on logon objects. A logon object Server object is obtained by issuing a RopLogon remote operation. Some properties are read-only. Some properties are write-only. Some properties can be deleted by the client. Some properties are available only on public folders logons. Some properties are available only on Private Mailbox logons. When the client attempts to read a non-readable property, write a non-writable

property, delete a non-deletable property, manipulate private-only properties on a public logon, or manipulate public-only properties on a private logon, the server MUST return ecPropSecurityViolation (0x80070005) for the property. See [MS-OXCROPS] for details about how error codes are returned for RopSetProps, RopGetProps, RopDeleteProperties.

To read any of the readable properties, issue a RopGetProps remote operation with a Server object of a logon obtained from a successful invocation of RopLogon. To write any of the writable properties, issue a RopSetProps remote operation with a Server object of a logon obtained from a successful invocation of RopLogon. To delete any of the deletable properties, issue a RopDeleteProperties remote operation with a Server object of a logon obtained from a successful invocation of RopLogon. See [MS-OXCROPS] for details of how to issue RopGetProps, RopSetProps, and RopDeleteProperties.

2.2.2.1 Private Mailbox Logon

The following properties are available on a Private Mailbox logon:

Name	Get	Set	Delete
PidTagExtendedRuleSizeLimit	X		
PidTagMaximumSubmitMessageSize	X		
PR_PROHIBIT_RECEIVE_QUOTA	X		
PidTagProhibitSendQuota	X		
PidTagStoreState	X		
PidTagComment	X	X	
PidTagContentCount	X		
PidTagDeleteAfterSubmit	X	X	X
PidTagDisplayName	X	X	
PidTagMailboxOwnerEntryid	X		
PidTagMailboxOwnerName	X		
PidTagMessageSize	X		
PidTagMessageSizeExtended	X		
PidTagOutOfOfficeState	X	X	
PidTagUserEntryid	X		
ptagSentMailSvrEID	X	X	X
PidTagCodePageId		X	
PidTagLocaleId		X	
PidTagSortLocaleId		X	

Name	Description

PidTagExtendedRuleSizeLimit	Maximum size, in bytes, the user is allowed to accumulate for a single "extended" rule. For details of extended rules, see [MS-OXORULE].
PidTagMaximumSubmitMessageSize	Maximum size, in kilobytes, of a message a user is allowed to submit for transmission to another user. An unset value, or a value of -1 means no limit.
PR_PROHIBIT_RECEIVE_QUOTA	Maximum size, in kilobytes, a user is allowed to accumulate in their mailbox, before no further mail will be delivered. An unset value, or a value of -1 means no limit.
PidTagProhibitSendQuota	Maximum size, in kilobytes, a user is allowed to accumulate in their mailbox, before the user can no longer submit any more mail. An unset value, or a value of -1 means no limit.
PidTagStoreState	See return value *StoreState* in *RopLogon* for full description.
PidTagComment	Mailbox comment.
PidTagContentCount	Cumulative count of normal (non-Associated) messages in the mailbox
PidTagDeleteAfterSubmit	Indicates if transport should delete all submitted mail after transmission. An unset value, or a value of FALSE means "no".
PidTagDisplayName	Mailbox display name
PidTagMailboxOwnerEntryid	The EntryId in the Global Address List of the owner of the mailbox
PidTagMailboxOwnerName	Display name of the owner of the mailbox.
PidTagMessageSize	Cumulative size, in bytes, of all content in the mailbox. Value is limited to 32 bits and becomes undefined if the content size exceeds four gigabytes.
PidTagMessageSizeExtended	Cumulative size, in bytes, of all content in the mailbox.
PidTagOutOfOfficeState	Indicates whether the user is Out of Office. Setting to FALSE causes accumulated OOF history maintained by the rules engine to be cleared for all folders/OOF rules. Setting to TRUE allows rules set to run only when the user is out of the office to run. See [MS-OXORULE].

PidTagUserEntryid	Address book entryID of the user logged on to the mailbox.
ptagSentMailSvrEID	SVREID structure identifying the Sent Items folder. Unset value means server won't move sent items to a Sent Items folder after transmission.
PidTagCodePageId	Establishes the client code page for Unicode ->DBCS string conversion. For details, see [MS-UCODEREF]
PidTagLocaleId	Establishes the language locale for translating system-generated messages, such as delivery reports. For details, see [MS-LCID]
PidTagSortLocaleId	Establishes the language locale for sorting the contents of tables. For details, see [MS-LCID]

2.2.2.2 Public Folders Logon

The following properties are available on a Public Folders logon.

Name	Get	Set	Delete
PidTagUserEntryid	X		
PidTagAddressBookMessageId	X		

Name	Description
PidTagUserEntryid	Address book entryID of the user logged on to public folders
PidTagAddressBookMessageId	Short-term MessageID of the first message in the local site's Offline Address Book public folder, if it exists and has a local replica. The property MUST have an error value of ecNotFound(0x8004010F) if there is no local site Offline Address Book public folder, the server can't open the folder, the server can't access msg, or there is no local replica of the folder.

3 Protocol Details

3.1 Client Details

None.

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.

The client MAY maintain a cached copy of the mapping between REPLIDs and REPLGUIDs used for Short and Long Term IDs.

The client MAY maintain a cache of per user data currently stored in the private store. This enables the client to only sync per user data when a change has been made.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Logging On To a Store

The client MUST log on a store using RopLogon before attempting to do any additional remote operations on the store.

When performing a RopLogon, the client MUST generate a LogonID to be used in the request (see [MS-OXCROPS] for more information on the ROP request). This LogonID is then used in all subsequent ROP requests which use this logon (see [MS-OXCROPS] for more information). This value MUST be unique per RPC Session Context Handle (calls to EcDoConnectEx). See [MS-OXCRPC] for more information.

If the server returns an error code of ecWrongServer, the client SHOULD re-attempt the logon using the correct server name supplied in the *Essdn* response field. This requires creating a new RPC connection to the server specified by *Essdn* (see [MS-OXCRPC] for information on creating the RPC connection) and re-attempting the logon using that connection.

If the server returns an error code of ecUnknownUser or ecLoginFailure, the client SHOULD use the Autodiscover HTTP Service protocol in order to attempt to retrieve updated user and server information. See [MS-OXDISCO] for more information on the Autodiscover HTTP Service protocol. If successful, the client can attempt to logon again by releasing the previous

RPC connection and creating a new RPC connection with the information supplied by the Autodiscover HTTP Service protocol. If the client is not successful at retrieving updated information or no changes are detected, the client MUST fail the logon.

If the client is unable to establish an RPC connection to a public store, it SHOULD request an alternative public folder store from the private store. The client can use an existing RPC connection to the private store, or create the new one. The client then executes a RopLogon against the private store. The client MUST set the ALTERNATE_SERVER flag in *OpenFlags*. The logon will return ecWrongServer and redirect the client to an alternate server. When performing the logon to the alternate server, the client MUST clear the ALTERNATIVE_SERVER flag and set the IGNORE_HOME_MDB flag in OpenFlags.

If the RPC session to the server is lost and then reconnected (see [MS-OXCRPC] for more details on RPC connections), the existing logon is invalid. The client MUST logon again by calling RopLogon (the client can reuse the existing LogonID). Additionally, all objects (folders, messages, tables,) opened using the original logon are now invalid and must be re-opened.

After logging on, the client SHOULD cache the *ReplGuid*. If a reconnect occurs, the new *ReplGuid* returned by RopLogon should be compared to the cached value. If the GUIDs are different, the client MUST dispose of all local caches of server information. This includes any open Server objects, caches of data mappings, or caches of special folder IDs. The effect MUST be similar to actually exiting the client application, and restarting from scratch.

3.1.4.2 Converting Between LongTerm and ShortTerm IDs

Short Term IDs use a 16-bit REPLID in place of a REPLGUID. Short Term IDs MUST NOT be persisted in any storage that could be accessed on a different logon session for the same mailbox. Any Short Term IDs cached in non-persistent storage MUST be forgotten (deleted from non-persistent storage) if the client reconnects to the server, issues a RopLogon, and the return value of *ReplGuid* is different than the value obtained from a previous RopLogon. To persist IDs in any long-term storage, the client MUST first convert the ID to a LongTermID. To convert between Short Term IDs and LongTermIDs (which uniquely identify an object globally), the client MUST use RopLongTermIDFromID and RopIDFromLongTermID.

3.1.4.3 Syncing Per User Read/Unread Data For Public Folders

Public folder data is replicated across multiple servers, with each server maintaining per-user read/unread data for each folder. The read/unread information is valid for that server only. If a subsequent logon results in the client being redirected to a different replica server, it is the client's responsibility to synchronize the current read/unread data to the new server.

For each folder, the client SHOULD issue a RopReadPerUserInformation against the public store (this is not necessary if the folder has not been modified) to retrieve the per-user read/unread data. This data is then stored in the private store using RopWritePerUserInformation. While the goal of these operations is to keep per-user data from the public store in sync with the private store, the actual time when this operation takes

place is up to the client (every time an item is marked read/unread, whenever a folder switch occurs, whenever the store is released).

When a folder is subsequently reopened in a later logon session, the client MUST check to see if the replica server has changed. This is done by issuing a RopGetPerUserGuid against the private store and comparing the *ReplGuid* returned with the public store's *ReplGuid* returned by RopLogon. If the REPLGUIDs match (or RopGetPerUserGuid doesn't find the REPLGUID), then the public folder is in sync. If the REPLGUIDs don't match, the client MUST synch the read/unread data from the private store up to the public store. This is done in the reverse manner as the previous sync: the data is retrieved from the private store by using RopReadPerUserInformation and sent to the public store using RopWritePerUserInformation.

When synching using RopReadPerUserInformation and RopWritePerUserInformation, it is important to note that the size of the return data potentially exceeds the maximum amount of data that can be communicated in a single remote operation. For this reason, the operation is designed to stream the data to the client by having the client invoke these remote operations multiple times. Because the server will cache interim data across client calls, the client MUST complete the entire streaming operation for the data of one folder before commencing streaming operations for another folder on the same server logon.

3.1.4.4 Registering For Notifications

The client can register for notifications for a store. For a full description of notifications, see [MS-OXCNOTIF].

3.1.5 Message Processing Events and Sequencing Rules

The client MUST log on a store using RopLogon before attempting to do any additional remote operations on the store.

3.1.5.1 ProcessingNotifications

For a description of notifications, see [MS-OXCNOTIF].

3.1.6 Timer Events

None

3.1.7 Other Local Events

None

3.2 Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not

mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The server will maintain several tables of data in order to satisfy the various remote operations that a client may invoke. These tables include: a REPLID and REPLGUID to-and-from mapping table, a mailbox table, a per-user data table, and a receive folder table.

The REPLID and REPLGUID to-and-from mapping table contains rows of 16-bit REPLID values coupled with 128-bit REPLGUID values. When a client invokes *RopIdFromLongTermId*, this table is searched for the REPLGUID portion of the ID passed by the client. If a row with that REPLGUID is found, the associated REPLID is used to formulate the returned short-term ID. If it is not found, a new row is added with the REPLGUID and a newly assigned REPLID, and that new REPLID is used to formulate the returned short-term ID. The newly assigned REPLID MUST be unique within the table. When a client invokes *RopLongTermIdFromId*, this table is searched for the REPLID portion of the ID passed by the client. If a row is found, the associated REPLGUID is used to formulate the return long-term ID. If a row is not found, the remote operation fails. Zero is an invalid REPLID value, and the server MUST NOT assign a REPLID with a value of zero.

The mailbox table is used for logging on to a private mailbox. The table contains one row for each mailbox in the database. It contains columns to specify the root and other special folders of the mailbox, the access permissions to the mailbox, an identifying GUID which matches the owner of the mailbox, along with other metadata, such as last logon time, various item counts and aggregate sizes within the mailbox, and so on. When a client invokes RopLogon, the client passes an identifying moniker for the mailbox. The server then looks up the moniker in a global directory. The entry in the global directory will indicate the proper server to log on to for this user's mailbox, and will contain other relevant data used to find the mailbox on that server, in the mailbox table. The proper row in the mailbox table is then found, and then the user's access is checked. If logon will be allowed, various special folder IDs are obtained from the table and returned to the client.

The per-user data table contains the read/unread information for various public folders on a specific public folder replica server. The table maintains the MailboxGUID, the REPLGUID, the Folder LongTermID and change number set of read items. The *RopGetPerUserLongTermIds, RopGetPerUserGuid, RopReadPerUserInformation, RopWritePerUserInformation* remote operations each make use of the data in this table.

The receive folder table contains rows of messages class strings and associated folder IDs. The delivery process will use the message class string on the incoming e-mail to look up the appropriate folder to which to deliver that message. The *RopGetReceiveFolder* and *RopSetReceiveFolder* remote operations each make use of the data in this table.

3.2.2 Timers

None.

3.2.3 Initialization

When a database is created, the database MUST be assigned a new randomly-generated REPLGUID.

When the REPLID and REPLGUID to-and-from mapping table is created, a single new entry MUST be added, consisting of the database REPLGUID and a newly assigned REPLID. When a database is restored from backup, the server MUST assign a new randomly-generated REPLGUID to the database, and add this new REPLGUID, along with a new REPLID, to the REPLID and REPLGUID to-and-from mapping table.

When a mailbox is created, the following entries MUST be added to the receive folder table for the new mailbox:

- "" (empty string) – Inbox in the new mailbox
- "IPM" – Inbox in the new mailbox
- "Report.IPM" – Inbox in the new mailbox
- "IPC" – root folder of the new mailbox

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

Except for RopLogon, all ROPs listed below have the client prerequisite of successfully completing a RopLogon operation. RopLogon requires the client has successfully connected to the server by initiating a normal RPC session (calls to EcDoConnectEx). See [MS-OXCRPC] for more information.

3.2.5.1 Processing RopLogon

If *OpenFlags* does not have the *PUBLIC* bit set, this logon is going to a private mailbox. Otherwise, this logon is going to the public folders.

3.2.5.1.1 Private Mailbox Logon

Look up the *UserEssdn* value in the global directory to get that user's configuration information. If lookup fails specifically because the *UserEssdn* could not be found, the server MUST fail the operation with a *ReturnCode* of 0x000003EB. Any other failures to find the user MUST fail the operation with a *ReturnCode* of 0x80040111.

If the user has no configured mailbox database, the remote operation MUST fail with a *ReturnCode* of 0x000003EB.

If the database indicated by the user's configured mailbox database is currently offline, the operation MUST fail with a *ReturnCode* of 0x80040111.

If the user's configured mailbox is not hosted by this server, the server MUST determine the name of the correct server hosting the user's mailbox and fail the remote operation with a *ReturnCode* of 0x00000478. See section 2.2.1.1.2.1.1 for details of properly forming the return values when a *ReturnCode* of 0x00000478 is sent.

If the user who has connected to the server is the same as the owner of the mailbox being logged on to, add the *TAKE_OWNERSHIP* flag to the *OpenFlags*fieldfield. This will be used in further processing.

If the *OpenFlags* field does not have the *USE_PER_MDB_REPLID_MAPPING* bit set, the server MUST identify a "reference database" to use. If this is the first logon operation being made on the current RPC session, the database opened on this logon will be the reference database. The server MUST then note in the RPC session state that this database is the reference database. If this is not the first logon operation being made on the current RPC session, the RPC session state will already have recorded in it the reference database to use. The server MUST use this database as the reference database for all REPLIDand REPLGUID to-and-from mapping table and named property mapping.

The server MUST verify the user logging on to the mailbox has permissions to it. First, the server MUST check if the user logging on has owner permission. If not, and the user logging on matches the owner of the mailbox, the remote operation MUST fail with a *ReturnCode* of 0x80070005. Next, the server MUST check if the user logging on has send-as permission. If the *OpenFlags* field has the *TAKE_OWNERSHIP* flag set and the user has owner permissions on the mailbox, the user will be granted full owner permission, send-as permission, and administrative "view" permission on the mailbox. Otherwise, the connecting user is considered a delegate. Delegate logons get send-as rights only if the permissions on the mailbox expressly grant the user those rights. Delegate logons do not get full ownership rights. Full owners of the mailbox are not required to have security settings checked before performing any non-administrative operations on the mailbox.

The server MUST then find the mailbox in the mailbox table. If the mailbox is not present in the table and the user has full owner rights on the mailbox, the server MUST create the mailbox. That process includes creating the default folders, establishing the proper receive folder values and so on. See section 3.2.3 for details. The server MUST NOT create the mailbox if the user hasn't got full owner permission. In that case, the remote operation MUST fail with a *ReturnCode* of 0x000003F2. Other failures to find the user in the mailbox table (beyond a "not found" error) MUST fail the operation with a *ReturnCode* of 0x80040111.

When creating a mailbox, if the *OpenFlags*fieldfield has the *NO_LOCALIZATION* flag set, the various special folders MAY be created with names in whatever language the server deems appropriate, but the mailbox MUST be marked for further localization.

If the mailbox is marked for further localization and the *OpenFlags* field has the *NO_LOCALIZATION* flag unset, and the user has full owner rights, the server MUST rename the various special folders (Inbox, Outbox,) to names appropriate for the system locale of the client.

The server MUST then determine the appropriate folder IDs to return to the user. See section 2.2.1.1.2.3.1 for details.

The server is now ready to accept further remote operation commands from the client on behalf of this logon session. The server should now update auditing information about the logon (last logon time, user identity, etc).

3.2.5.1.2 Public Folders Logon

The server MUST confirm the user logging on to the public folders has a mailbox in the organization. The server MUST perform the following:

1. Determine the mailbox database hosting the connected user's mailbox. The user is determined from the underlying RPC connection. The *UserEssdn* field specifies a mailbox to log on to for private mailbox logons only. This field will be empty for public folder logons.
2. Determine from the global configuration for that mailbox database the preferred public folder database to use.
3. Determine the server that database is hosted upon.

If the *OpenFlags* field has the ALTERNATIVE_*SERVER* bit set, the server MUST search for another public folder database server in the organization. The process by which one is chosen is up to the implementation, however, the server SHOULD choose a public folder database server that is not the configured preferred server. The server MAY choose the configured preferred public folder server. If a suitable server cannot be found, the operation MUST fail with a *ReturnCode* of 0x80040111. Otherwise, the operation MUST fail with a *ReturnCode* of 0x00000478. See section 2.2.1.1.2.1.1 for details of properly forming the return values when a *ReturnCode* of 0x00000478 is sent.

If the server being queried does not host the preferred public folder database and the *LogonFlags* field does not have the *Ghosted* bit set and the *OpenFlags* field does not have the *IGNORE_HOME_MDB* bit set, the operation MUST fail with a *ReturnCode* of 0x00000478. See section 2.2.1.1.2.1.1 for details of properly forming the return values when a *ReturnCode* of 0x00000478 is sent.

If this server doesn't host a public folder database at all, or the database is not presently accessible, the operation MUST fail with a *ReturnCode* of 0x80040111.

The server MUST then determine the appropriate folder IDs to return to the user. See section 2.2.1.1.2.3.2 for details.

The server is now ready to accept further remote operation commands from the client on behalf of this logon session.

3.2.5.2 Processing RopGetReceiveFolder

The server MUST verify the operation is being performed against a private mailbox logon, and not a public folders logon. If the operation is performed against a public folders logon, the server MUST fail the operation with a *ReturnCode* of 0x80040102.

The server MUST validate the *MessageClass* field as described in section 2.2.1.2.1.1. If the *MessageClass* field does not conform to the requirements, the server MUST fail the operation with a *ReturnCode* of 0x80070057.

The server then MUST scan the receive folder table to find the row which has the longest prefix string, case-insensitive, match to *MessageClass*. The search MUST be scoped to the relevant mailbox. The server then MUST retrieve the actual message class string from the table, and the associated folder ID.

If no entry in the table can be matched, the server MUST fail the operation with a *ReturnCode* of 0x00000463.

Upon success, the server returns the actual configured message class string and the folder ID. The server MAY case-fold the string to all upper case, all lower case, or leave the string as stored.

3.2.5.3 Processing RopSetReceiveFolder

The server MUST verify the operation is being performed against a private mailbox logon, and not a public folders logon. If the operation is performed against a public folders logon, the server MUST fail the operation with a *ReturnCode* of 0x80040102.

The server MUST validate the *MessageClass* field as described in section 2.2.1.3.1.2. If the *MessageClass* field does not conform to the requirements, the server MUST fail the operation with a *ReturnCode* of 0x80070057.

If the *MessageClass* field is a case-insensitive match to either “IPM” or “Report.IPM”, the server MUST fail the operation with a *ReturnCode* of 0x80070005.

If the *MessageClass* field is a zero-length string, or *FolderID* field is all zeros, the server MUST fail the operation with a *ReturnCode* of 0x80004005.

The server MUST search the receive folder table using case-insensitive string compare for an exact match to the *MessageClass* field fieldThe search MUST be scoped to the relevant mailbox. If a match is found, the value of the *FolderID* field MUST replace the folder ID stored in the table on that row. If a match is not found, a new row is added with the *MessageClass* and *FolderID* field values. The server MAY case-fold the *MessageClass* value to upper case, to lower case or leave the value unchanged before storage. After modifying or inserting the new row, the *LastModificationTime* column for that row MUST be set to the current system time of the server, adjusted to UTC.

3.2.5.4 Processing RopGetReceiveFolderTable

The server MUST verify the operation is being performed against a private mailbox logon, and not a public folders logon. If the operation is performed against a public folders logon, the server MUST fail the operation with a *ReturnCode* of 0x80040102.

The server MUST return all entries of the Receive Folder Table for the relevant mailbox. The server MAY coerce *MessageClass* values to all upper case or all lower case or return the value as stored.

3.2.5.5 Processing RopGetStoreState

The server MUST verify the operation is being performed against a private mailbox logon, and not a public folders logon. If the operation is performed against a public folders logon, the server MUST fail the operation with a *ReturnCode* of 0x80040102.

If the mailbox has any persisted search folders, the server MUST set the *STORE_HAS_SEARCHES* flag in the return value.If the mailbox does not have any persisted search folders, the server MUST NOT set the *STORE_HAS_SEARCHES* flag in the return value.

The server MUST NOT set any other flags in the return value.

3.2.5.6 Processing RopGetOwningServers

The server SHOULD verify the operation is being performed against a public folders logon. If the operation is performed against a private mailbox store, the server MAY fail the operation, or it MAY compute a correct answer for the client.

Each public folder has associated configuration information, including which servers are configured to actually hold content of the folder. This specific configuration indicates one of several potential states for each replica server. An "Active" replica contains content and is expected to serve that content to clients. An "Inactive" replica contains content and is not going to serve that content to clients. A "Deleted" replica once contained content and presently does not. There are other states as well, all of which result in the server not serving content to clients.

Each server in the organization's network has a tangible communication cost due to network hardware costs, the cost of the network connectors (various WAN vs LAN costs and so forth), perceived cost of using the network for certain applications, and so on.

The server MUST retrieve the current replica info for the specific public folder specified by the *FolderID* field. This replica info is a list, each entry consisting of at least a server identifier and the replication state for this folder on that server (Active/Inactive/Deleted/etc). The server MUST obtain the network cost for each server in the list, if that cost information isn't already in the list. The network cost values are expressed relative to the server servicing the request, not the client making the request.

The server MUST remove entries from the list which are not Active replicas.The server MAY remove entries from the list which, at its discretion, it determines to be too expensive for the client to reach. The server MAY remove entries that other configuration indicates that the client should be prohibited from attempting a connection, if such configuration exists. If the resulting trimmed list is empty, the operation MUST fail with a *ReturnCode* of 0x00000469.

The server MUST sort the list according to the cost information, least expensive items sorting to the front of the list. Servers with the same cost MAY appear in any order, but the server SHOULD ensure that the same list values sort to the same order every time.

The server MUST count the number of entries at the front of the list which all have the same cost value. The resultant value will be the number of cheapest, equally costed servers (the *CheapServersCount* return value).

The current total list length constitutes the *OwningServersCount* return value. The list contents of server identifiers constitutes the *Servers* return value. The server MUST map whatever identifier moniker for each server it has into an ESSDN string to return to the client.

3.2.5.7 Processing RopPublicFolderIsGhosted

The server SHOULD verify the operation is being performed against a public folders logon. If the operation is performed against a private mailbox store, the server MAY fail the operation, or it MAY compute a correct answer for the client.

Each public folder has associated configuration information, including which servers are configured to actually hold content of the folder. This specific configuration indicates one of several potential states for each replica server. An "Active" replica contains content and is expected to serve that content to clients. An "Inactive" replica contains content and is not going to serve that content to clients. A "Deleted" replica once contained content and presently does not. There are other states as well, all of which result in the server not serving content to clients.

Each server in the organization's network has a tangible communication cost due to network hardware costs, the cost of the network connectors (various WAN vs LAN costs, and so forth), perceived cost of using the network for certain applications, and so on.

The server MUST retrieve the current replica information for the specific public folder specified by the *FolderID* field. This replica information is a list, each entry consisting of at least a server identifier and the replication state (Active, Inactive, Deleted) for this folder on that server . The server MUST obtain the network cost for each server in the list, if that cost information isn't already in the list. The network cost values are expressed relative to the server, not the client making the request.

The server MUST return a *GhostedState* value of TRUE if the queried server is not listed as an Active replica of the folder. The *GhostedState* value MUST be FALSE if the queried server is listed as an Active replica of the folder.

The server MUST remove entries from the list which are not Active replicas. The server MAY remove entries from the list which, at its discretion, it determines to be too expensive for the client to reach. The server MAY remove entries that other configuration indicates that the client should be prohibited from attempting a connection, if such configuration exists. If the resulting trimmed list is empty, the operation must be failed with a *ReturnCode* of 0x00000469. A client MUST interpret this *ReturnCode* value as implying a *GhostedState* value of true.

The server MUST sort the list according to the cost information, least expensive items sorting to the front of the list. Servers with the same cost MAY appear in any order, but the server SHOULD ensure that the same list values sort to the same order every time.

The server MUST count the number of entries at the front of the list which all have the same cost value. The resultant value will be the number of cheapest, equally costed servers (the *CheapServersCount* return value).

The current total list length constitutes the *OwningServersCount* return value. The list contents of server identifiers constitutes the *Servers* return value. The server MUST map whatever identifier moniker for each server it has into an ESSDN string to return to the client.

3.2.5.8 Processing RopLongTermIdFromId

The server MUST verify the REPLID portion of the given ID exists in the REPLID and REPLGUID to-and-from mapping table. The server MUST NOT otherwise attempt to validate the ID.

If the given REPLID is not in the REPLID and REPLGUID to-and-from mapping table, the operation must fail with 0x8004010F.

The server MUST construct the returned LongTermID by composing the REPLGUID recovered from the REPLID and REPLGUID to-and-from mapping table for the REPLID portion of the given ID, copying the Global Counter portion of the given ID to the Global Counter portion of the LongTermID, and by setting the padding bytes of the LongTermID to zero.

3.2.5.9 Processing RopIdFromLongTermId

The server MUST NOT perform any specific validation of the passed LongTermID value. The server MUST search the REPLID and REPLGUID to-and-from mapping table for the REPLGUID portion of the LongTermID field. If a row is not found, the server MUST add a new row consisting of the REPLGUID portion of the LongTermID field and a newly assigned REPLID value. The new REPLID value MUST be unique in the REPLID and REPLGUID to-and-from mapping table.

The server MUST construct the returned ID by composing the REPLID recovered from the REPLID and REPLGUID to-and-from mapping table for the REPLGUID portion of the LongTermID, and by copying the Global Counter portion of the LongTermID to the returned ID.

The server MUST ignore the content of the padding bytes in the LongTermID field.

3.2.5.10 Processing RopGetPerUserLongTermIds

The server MUST verify the operation is being performed against a private mailbox logon, and not a public folders logon. If the operation is performed against a public folders logon, the server MUST fail the operation with a *ReturnCode* of 0x80040102.

The server MUST search the Per User Data table for the mailbox for entries identified by the *ReplGuid*fieldfield. For each entry in the table, the server MUST collect the associated public folder LongTermID. The total number of LongTermIDs collected constitutes the *IDCount* return value, the aggregated list of LongTermIDs constitutes the *IDList* return value.

The server MAY return the list of LongTermIDs in any order. The server MAY return an empty list.

3.2.5.11 Processing RopGetPerUserGuid

The server MUST verify the operation is being performed against a private mailbox logon, and not a public folders logon. If the operation is performed against a public folders logon, the server MUST fail the operation with a *ReturnCode* of 0x80040102.

The server MUST search the Per User Data table for the mailbox for the only row with a folder ID equal to the *FolderLongTermID* field. The server MUST return the associated REPLGUID value in the *ReplGuid* return value.

3.2.5.12 Processing RopReadPerUserInformation

This operation MAY be issued against either a private mailbox logon, or a public folders logon. Specific behavior is specified below.

3.2.5.12.1 Private Mailbox Specific Behavior

The server MUST search the Per User Data table for the mailbox for the only row with a folder ID equal to the *FolderID* field. The server MUST retrieve from that row the stored change number set of read items. The change number set MUST be serialized as a Serialized IDSET with REPLGUID as specified in [MS-OXCFXICS]. The server then MUST return to the client that portion of the resulting binary BLOB indicated by the *DataOffset* and *MaxDataSize* fields. Specifically, the first byte of the BLOB to return to the client is at offset *DataOffset*. The number of bytes to return MUST NOT exceed *MaxDataSize*.

The return value of *HasFinished* MUST be TRUE if *DataOffset*+ *DataSize* equals the total size of the BLOB. The return value of *DataSize* MUST be the minimum of the amount of data actually read from the database, *MaxDataSize* and(the total size of the BLOB minus *DataOffset*).

3.2.5.12.2 Public Folders Specific Behavior

The server MUST first determine that the persisted read/unread information for the user is up to date. If the server is maintaining any in-memory caches of the per-user read/unread information, the data for the current user MUST now be flushed to disk. The server MAY choose to flush this data to disk at other opportune times.

The server MUST search the Per User Data table for the only row with a folder ID equal to the *FolderID* field and the user ID equal to the logged on user. The server MUST retrieve from that row the stored change number set of read items. The change number set MUST be serialized as a Serialized IDSET with REPLGUID as specified in [MS_OXCFXICS]. The server then MUST return to the client that portion of the resulting binary BLOB indicated by the *DataOffset* and *MaxDataSize* fields. Specifically, the first byte of the BLOB to return to the client is at offset *DataOffset*. The number of bytes to return MUST NOT exceed*MaxDataSize*.

The return value of *HasFinished* MUST be TRUE if *DataOffset*+ *DataSize* equals the total size of the BLOB. The return value of *DataSize* MUSTbe the minimum of the amount of data actually read from the database, *MaxDataSize* and (the total size of the BLOB minus *DataOffset*).

3.2.5.13 Processing RopWritePerUserInformation

This operation MAY be issued against either a private mailbox logon, or a public folders logon. Specific behavior is specified below.

3.2.5.13.1 Common Behavior

Each invocation of this remote operation accumulates data from the client until the client makes a final call with *HasFinished* set to TRUE. The server MUST maintain proper bookkeeping to ensure the data accumulates completely before anything is persisted to storage.

The server MUST determine if the current invocation is a continuation of a previous invocation by examining the *FolderID* and *DataOffset* fields. If the folder ID has changed since the last invocation, or the *DataOffset* value does not equal the amount of data already written, the server MUST assume the previous operation was aborted and MUST dispose any accumulated data. In addition, if the current invocation's *DataOffset* isn't zero, the remote operation MUST fail with a *ReturnCode* of 0x80004005.

Once the client invokes this remote operation with *HasFinished* set to TRUE, the server MUST validate the accumulated data and verify it is a properly formed serialized IDSET with REPLGUID as specified in [MS-OXCFXICS]. If the data is not properly formed, the remote operation MUST fail with a *ReturnCode* of 0x000004ED.

After performing the specific behavior below, the server MUST record, in UTC, the current system time on the appropriate row in the table.

3.2.5.13.2 Private Mailbox Specific Behavior

The server MUST search the Per User Data table for the mailbox for the only row with a folder ID equal to the *FolderID* field. If the row exists, the *ReplGuid* field and accumulated change number information MUST replace any existing values in the table. If the row does not presently exist, a new row for the mailbox and folder MUST be added, setting the *ReplGuid* field and accumulated change number information onto that row.

3.2.5.13.3 Public Folders Specific Behavior

The server MUST search the Per User Data table for the only row with a user ID equal to user ID associated with the session logon and a folder ID equal to the *FolderID* field. If the row exists, the accumulated change number information MUST replace any existing values in the table. If the row does not presently exist, a new row for the user and folder MUST be added, setting the accumulated change number information onto that row.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

Here is a sample sequence of ROP requests and responses that a client and a server might exchange. 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 specification in [MS-OXCROPS]. These examples assume the client has already successfully connected to the server. For more details, see [MS-OXCROPS].

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 bold value at the far left 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 underlines byte "4d" (01001101) is located 0x83 bytes (131 bytes) from the beginning of the buffer. The dash between eighth byte ("44") and ninth byte ("4c") bytes 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.

The following example shows how a "Property Tag" and its "Property Value" are represented in a buffer and interpreted directly from it (according to the PropertyValue structure format specified in [MS-OXCDATA]). The data appears in the buffer in little-endian format.

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

Property Tag: 0x66760003 (PidTagRuleSequence)

Property Value: 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 is ULONG.

4.1 RopLogon for a private mailbox

Request:

```
0000: 01 0C 04 00 01 00 00 00-00 68 00 2F 6F 3D 46 69
```

0010: 72 73 74 20 4F 72 67 61-6E 69 7A 61 74 69 6F 6E

0020: 2F 6F 75 3D 45 78 63 68-61 6E 67 65 20 41 64 6D

0030: 69 6E 69 73 74 72 61 74-69 76 65 20 47 72 6F 75

0040: 70 20 28 46 59 44 49 42-4F 48 46 32 33 53 50 44

0050: 4C 54 29 2F 63 6E 3D 52-65 63 69 70 69 65 6E 74

0060: 73 2F 63 6E 3D 41 64 6D-69 6E 69 73 74 72 61 74

0070: 6F 72 00

[0000-0000] LogonFlags - *Private*

[0001-0004] OpenFlags -*HOME_LOGON, TAKE_OWNERSHIP, NO_MAIL, USE_PER_MDB_REPLID_MAPPING*

[0005-0008] StoreState (value is ignored by the server)

[0009-000A] Size - the following string is 0x68 bytes long

[000B-0072] UserEssdn

Response:

0000: 01 01 00 00 00 00 78 27-1A 01 00 00 00 00 78 27

0010: 1C 01 00 00 00 00 78 27-1D 01 00 00 00 00 78 27

0020: 1B 01 00 00 00 00 78 27-1E 01 00 00 00 00 78 27

0030: 1F 01 00 00 00 00 78 27-20 01 00 00 00 00 78 27

0040: 21 01 00 00 00 00 78 27-24 01 00 00 00 00 78 27

0050: 25 01 00 00 00 00 78 27-22 01 00 00 00 00 78 27

0060: 23 01 00 00 00 00 78 27-26 07 F7 F8 91 A5 1C 34

0070: 16 41 8C 48 9D B0 1A 86-F5 0B 01 00 4D 77 D4 64

0080: 83 49 70 4F 9B 8B 46 E6-35 BB 78 AB 0D 10 0F 01

0090: 0A 03 D8 07 60 53 1A C2-BE 82 C8 01 00 00 00 01

[0000-0000] LogonFlags - *Private*

[0001-0008] Mailbox Root Folder ID

[0009-0010] Deferred Action Folder

[0011-0068] < more folder IDs >

[0069-0069] Flags - *SendAsRight, OwnerRight, Localized*

[006A-0079] MailboxGuid

[007A-007B] ReplId

[007C-008B] ReplGuid

[008C-0093] LogonTime - 2008/03/10 Mon 15:10:13

[0094-009B] GWARTTime - 2008/03/10 14:55:19

[009C-009F] StoreState - *STORE_HAS_SEARCHES*

4.2 RopLogon for public folders

Request:

```
0000: 00 06 04 00 01 00 00 00-00 00 00
```

[0000-0000] LogonFlags - *Public*

[0001-0004] OpenFlags -*PUBLIC*, *HOME_LOGON, TAKE_OWNERSHIP, USE_PER_MDB_REPLID_MAPPING*

[0005-0008] StoreState (value is ignored)

[0009-000A] Size (no mailbox legacy DN is given for public logons)

Response:

```
0000: 00 01 00 00 00 00 00 00-06 01 00 00 00 00 00 00
0010: 01 01 00 00 00 00 00 00-02 01 00 00 00 00 00 00
0020: 03 01 00 00 00 00 00 00-04 01 00 00 00 00 00 00
0030: 05 00 00 00 00 00 00 00-00 03 00 00 00 00 00 00
0040: 07 03 00 00 00 00 00 00-08 00 00 00 00 00 00 00
0050: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0060: 00 00 00 00 00 00 00 00-00 01 00 70 5B CA BF 1E
0070: F9 98 41 89 7D 47 9E 09-45 FD 2F 95 DA FE 74 E0
0080: F7 4C 4C 81 EF 83 BA 85-0B E8 E4
```

[0000-0000] LogonFlags - *Public*

[0001-0008] Public Folders Root Folder ID

[0009-0010] IPM subtree root folder ID

[0011-0050] Other folder IDs

[0051-0068] Unused

[0069-006A] ReplId

[006B-007A] ReplGuid

[007B-008A] PerUserGuid

4.3 RopGetReceiveFolder

Request:

0000: 00

[0000-0000] MessageClass<empty string>

Response:

0000: 01 00 00 00 00 78 27 1E-00

[0000-0007] FolderID

[0008-0008] ExplicitClass<empty string>

4.4 RopIdFromLongTermId

Request:

0000: 70 5B CA BF 1E F9 98 41-89 7D 47 9E 09 45 FD 2F

0010: 00 00 00 00 00 12 00 00

[0000-000F] LongTermIDREPLGUID

[0010-0015] LongTermID Counter

[0016-0017] LongTermID padding

Response:

0000: 05 00 00 00 00 00 00 12

[0000-0001] Short-Term ID Replid

[0002-0007] Short-Term ID Counter

4.5 RopReadPerUserInformation

Request:

0000: 70 5B CA BF 1E F9 98 41-89 7D 47 9E 09 45 FD 2F

0010: 00 00 00 00 00 12 00 00-00 00 00 00 00 00 00 00

[0000-0017] FolderLongTermID

[0018-0018] SendOnlyIfChanged

[0019-001C] ReadOffset

[001D-001E] ReadAmount

Response:

0000: 01 18 00 D8 44 AE 73 F9-61 5D 4F B3 C6 9A 7C 31

0010: FE C1 23 06 00 00 00 78-2B 33 00

[0000-0000] LastChunk

[0001-0002] Size

[0003-001A] ChunkData

4.6 RopWritePerUserInformation

Request:

```
0000: 70 5B CA BF 1E F9 98 41-89 7D 47 9E 09 45 FD 2F
0010: 00 00 00 00 00 12 00 00-01 00 00 00 00 18 00 D8
0020: 44 AE 73 F9 61 5D 4F B3-C6 9A 7C 31 FE C1 23 06
0030: 00 00 00 78 2B 33 00 D8-44 AE 73 F9 61 5D 4F B3
0040: C6 9A 7C 31 FE C1 23
```

[0000-0017] FolderLongTermID

[0018-0018] LastChunk

[0019-001C] WriteOffset

[001D-001E] ChunkSize

[001F-0036] ChunkData

[0037-0046]ReplGuid

Response:

<no response>

4.7 RopGetPerUserLongTermIds

Request:

```
0000: 4D 77 D4 64 83 49 70 4F-9B 8B 46 E6 35 BB 78 AB
```

[0000-000F] ReplGuid

Response:

00 00

[0000-0001] IDCount (no IDs being returned)

4.8 RopGetReceiveFolderTable

Request:

No fields in the request

Response:

```
0000: 04 00 00 00 00 01 00 00-00 00 78 27 1E 00 5E FF
0010: 54 5F C0 82 C8 01 00 01-00 00 00 00 78 27 1A 49
0020: 50 43 00 32 EF 56 5F C0-82 C8 01 00 01 00 00 00
```

0030: 00 78 27 1E 49 50 4D 00-32 EF 56 5F C0 82 C8 01

0040: 00 01 00 00 00 00 78 27-1E 52 45 50 4F 52 54 2E

0050: 49 50 4D 00 32 EF 56 5F-C0 82 C8 01

[0000-0003] RowCount (4 rows being returned)

[0004-0004], [0016-0016], [002B-002B], [0040-0040] Error Flag Indicator (no error)

[0005-000C], [0017-001E], [002C-0033], [0041-0048] FolderID

[000D-000D], [001F-0022], [0034-0037], [0049-0053] MessageClass

[000E-0015], [0023-002A], [0038-003F], [0054-005B] LastModificationTime

4.9 RopSetReceiveFolder

Request:

0000: 01 00 00 00 00 78 27 1A-49 50 4D 2E 53 6F 6D 65

0010: 4D 65 73 73 61 67 65 43-6C 61 73 73 00

[0000-0007] FolderID

[0008-001C] MessageClass

Response:

No response

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the Store Object Protocol. General security considerations pertaining to the underlying RPC-based transport apply. For details, see [MS-OXCROPS].

5.2 Index of Security Fields

None.

6 Appendix A: Microsoft Office Outlook and Microsoft Exchange Behavior

The information in this specification is applicable to the following versions of Office/Exchange:

- Office 2003 with Service Pack 3 applied
- Exchange 2003 with Service Pack 2 applied
- Office 2007 with Service Pack 1 applied

- Exchange 2007 with Service Pack 1 applied

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Office/Exchange behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies Office/Exchange does not follow the prescription.

<1> Microsoft Exchange Server maintains a single REPLID and REPLGUID to-and-from mapping table per database. There is no per-mailbox scoping of data.

<2> Microsoft Exchange Server assigns REPLIDs in increasing sequential order, starting with the number 1.

<3> Microsoft Exchange Server maintains a single receive folder table per database. The data within the table are scoped to each mailbox by including a per-mailbox moniker on each row. Conceptually, there is a single receive folder table per mailbox.

<4> Microsoft Exchange Server case-folds all message class strings in the Receive Folder table to upper case. All message class strings returned from RopGetReceiveFolder and RopGetReceiveFolderTable are sent to the client in upper case.

<5> Microsoft Exchange Server may successfully complete a RopGetOwningServers operation when issued against a private mailbox logon. The results are undefined.

<6> Microsoft Exchange Server will return ecServerPaused after five attempts within any 10 second period to log on to a mailbox which is not hosted on the server.

<7> Microsoft Exchange Server 2007 implements logic to interpret the *USE_PER_MDB_REPLID_MAPPING* logon flag. All versions prior to Microsoft Exchange Server 2007 ignore this logon flag and operate as if it hadn't been set.

<8> Microsoft Office Outlook caches the *ReplGuid* returned by RopLogon. If a reconnect occurs and the *ReplGuid* changes, Outlook fails the logon and prompts the user to restart the application.

<9> If the user doesn't exist in the Active Directory forest, Exchange 2007 returns ecUnknownUser, but Exchange 2003 returns ecLoginFailure.

<10> Autodiscover HTTP Service protocol is only supported in Outlook 2007.

Index