[MS-OXCFXICS]:
Bulk Data Transfer Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Revised and edited technical content.
12/03/2008	1.03		Revised and edited technical content.
02/04/2009	1.04		Revised and edited technical content.
03/04/2009	1.05		Editorial updates.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	5.0.0	Major	Updated and revised the technical content.
05/05/2010	6.0.0	Major	Updated and revised the technical content.
08/04/2010	7.0	Major	Significantly changed the technical content.
11/03/2010	8.0	Major	Significantly changed the technical content.
03/18/2011	9.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Bulk Data Transfer Protocol enables the bulk transmission of **mailbox** data, represented by folders and messages, between clients and servers. This protocol is commonly used for replicating, exporting, or importing mailbox content between clients and servers.

This document specifies the following:

- How a client can configure a **ROP** to upload a set of folders or messages to a server, or download a set of folders or messages from a server.
- How a client or a server can receive and reconstitute folders and messages that are transmitted from another client or another server.
- How a client can upload changes made to local folders and message replicas to a server.
- Semantics of ROPs that are used to fulfill the aforementioned operations.

Sections 1.8, 2, and 3 of this specification are normative and contain RFC 2119 language. Sections 1.5 and 1.9 are also normative but cannot contain RFC 2119 language. All other sections and examples in this specification are informative.

1.1 Glossary

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

ASCII
Augmented Backus-Naur Form (ABNF)
code page
flags
GUID
handle
little-endian
property set
remote procedure call (RPC)
Unicode

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

Attachment object
condition
content synchronization
Deleted Items folder
Embedded Message object
enterprise/site/server distinguished name (ESSDN)
EntryID
external identifier
FastTransfer download context
FastTransfer stream
FastTransfer upload context
folder associated information (FAI)
Folder object
foreign identifier
ghosted
global identifier
hard delete

hierarchy synchronization
Incremental Change Synchronization (ICS)
internal identifier
interpersonal messaging subtree
local replica
LongTermID
mailbox
marker
message body
Message object
messaging object
named property
normal message
offline
Outbox folder
property ID
property tag
property type
public folder
recipient
Recipient object
remote operation (ROP)
replica
replica GUID (REPLGUID)
replica ID (REPLID)
Rich Text Format (RTF)
ROP request buffer
ROP response buffer
Sent Items folder
server replica
soft delete
store
synchronization context
synchronization download context
synchronization scope
synchronization type
synchronization upload context
top-level message

The following terms are specific to this document:

base property type: The type of the property, if the property is single-valued, or the type of an element of the property, if the property is multi-valued.

camel-cased: The capitalization style applied to compound words or phrases when they are joined without spaces and the first letter of each word, except the first word, is capitalized within the compound. For example, displayName is camel-cased.

change number: A number that identifies a version of a messaging object. A change number is identical in format to a message ID (MID) or folder ID (FID).

checkpoint ICS state: An Incremental Change Synchronization (ICS) state that is provided by a server in the middle of an ICS operation, which reflects the state of the local replica, indicated by the initial ICS state, after applying all differences transmitted in the ICS operation.

common byte stack: A list of arrays of bytes. Byte values of contained arrays, when together in their natural order, represent common high-order bytes of GLOBCNT values. Common byte stacks are used in a last-in first-out (LIFO) fashion during serialization or deserialization of GLOBSETs.

conflict detection: A process that is used to determine whether two versions of the same object conflict with each other, that is, one is not a direct or indirect predecessor of another.

conflict handling: One or more actions that are taken upon detection of a conflict between versions of the same object. These actions include conflict reporting and conflict resolution.

conflict reporting: An automated process that notifies a system actor of a previously detected conflict.

conflict resolution: An automated or semi-automated process that is used to resolve a previously detected conflict between versions of an object. The process replaces conflicting versions with a successor version. How a successor version relates to a conflicting version depends on the algorithm that is used.

deleted item list: An abstract repository of information about deleted items.

expired Message object: A Message object that was removed by a server due to the age of the Message object.

FastTransfer context: Either a FastTransfer download context or a FastTransfer upload context.

final ICS state: An Incremental Change Synchronization (ICS) state that is provided by a server upon completion of an ICS operation. A final ICS state is a checkpoint ICS state that is provided at the end of the ICS operation.

initial ICS state: An Incremental Change Synchronization (ICS) state that is provided by a client when it configures an ICS operation.

meta-property: An entity that is identified with a property tag containing information (a value) that describes how to process other data in a FastTransfer stream.

partial completion: The outcome of a complex operation with independent steps, where some steps succeeded and some steps failed.

Pascal-cased: The capitalization style applied to compound words or phrases when they are joined without spaces and the first letter of each word (including the first word) is capitalized within the compound. For example, DisplayName is a pascal-cased.

Predecessor Change List (PCL): A set of change numbers that specify the latest versions of a messaging object in all replicas that were integrated into the current version. It is used for conflict detection.

property list restriction table: A set of restrictions, expressed in tabular form, that is imposed on an array of properties and the values of those properties.

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

1.2 References

1.2.1 Normative References

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

[MS-OXBBODY] Microsoft Corporation, "Best Body Retrieval Protocol Specification", June 2008.

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

[MS-OXCFOLD] Microsoft Corporation, "Folder Object Protocol Specification", June 2008.

[MS-OXCMSG] Microsoft Corporation, "Message and Attachment Object Protocol Specification", June 2008.

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

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

[MS-OXCSTOR] Microsoft Corporation, "Store Object Protocol Specification", June 2008.

[MS-OXCSYNC] Microsoft Corporation, "Mailbox Synchronization Protocol Specification", June 2008.

[MS-OXOMSG] Microsoft Corporation, "E-Mail Object Protocol Specification", June 2008.

[MS-OXPROPS] Microsoft Corporation, "Exchange Server Protocols Master Property List", April 2008.

[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

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.ietf.org/rfc/rfc5234.txt

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-OXCMAIL] Microsoft Corporation, "RFC2822 and MIME to E-Mail Object Conversion Protocol Specification", June 2008.

[MS-OXCPRPT] Microsoft Corporation, "Property and Stream Object Protocol Specification", June 2008.

[MS-OXCSPAM] Microsoft Corporation, "Spam Confidence Level Protocol Specification", June 2008.

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

[MS-OXOABK] Microsoft Corporation, "Address Book Object Protocol Specification", April 2008.

[MS-OXOCAL] Microsoft Corporation, "Appointment and Meeting Object Protocol Specification", June 2008.

[MS-OXOFLAG] Microsoft Corporation, "Informational Flagging Protocol Specification", June 2008.

[MS-OXORMDR] Microsoft Corporation, "Reminder Settings Protocol Specification", June 2008.

[MS-OXOTASK] Microsoft Corporation, "Task-Related Objects Protocol Specification", June 2008.

1.3 Overview

This document specifies how clients and servers can efficiently exchange data that is represented as folders and messages that are contained in private or public mailboxes.

Efficiency in the exchange of data is achieved through the following means:

- Packaging data for several folders or messages into a single ROP response, which can be compressed at the **remote procedure call (RPC)** level.
- Reducing transmitted data to only changes that the user is interested in.
- Reducing transmitted data to only changes that relate to a subset of folder or message data by using **Incremental Change Synchronization (ICS)**.
- Performing optimizations on the server provided that the server knows the scope of the operation ahead of time.
- Minimizing the bandwidth required to copy message and folder content by efficiently packing data by using **FastTransfer streams**.

This document supports the transfer of data in scenarios that derive from the following semi-independent variables:

1. Direction of data transmission: download or upload.
2. Type of **messaging objects** included in a transmission: folders, messages, or both.
3. Scope of the data that is transmitted for a messaging object. The scope might be one of the following:
   - A full object or a subset of its data
   - Changes since the last transmission
   - Operations such as a read state change or a move
4. Scope of the messaging objects that are included in a set. The scope might be one of the following:
   - Identified directly by **Folder object** identifiers and **Message object** identifiers
   - Identified by a combination of criteria and state information maintained by the client

This specification is based on the following roles: one server, and one or more clients. The only exception is the server-to-client-to-server upload scenario described in section 3.3.5.4.2.1.

1.3.1 FastTransfer Copy Operations

FastTransfer copy operations enable clients to efficiently copy the content of explicitly specified folders, messages, and attachments between replicas of the same or different mailboxes by using a special binary format known as a FastTransfer stream as the medium. A FastTransfer stream contains copies of folder, message, or attachment content in a predefined serialized format. The

FastTransfer stream can be used to create copies of this folder, message, or attachment content in any destination folder, on any mailbox, on any client, or on any server.

Every FastTransfer operation is independent. After the operation is complete, no state has to be maintained on the client or on the server.

FastTransfer download operations enable clients to download a copy of the explicitly specified folders, messages, or attachments in the FastTransfer stream format. The resulting FastTransfer stream can be either interpreted on the client, or used in a FastTransfer upload operation if the intent is to copy messaging objects between mailboxes on different servers.

FastTransfer upload operations enable a client to create new folders or modify content of existing folders, messages, and attachments by encoding data into the FastTransfer stream format.

1.3.2 Incremental Change Synchronization

ICS enables servers and clients to keep synchronized versions of messages, folders, and their related properties on both systems. Changes that are made to messages and folders on the client are replicated to the server and vice versa. ICS can determine differences between two folder hierarchies or two sets of content, and can upload or download information about the differences in a single session.

Changes to folder properties, changes to the folder hierarchy, and folder creations and deletions are included in **hierarchy synchronization** operations.

Changes to message properties, changes to read and unread message state, changes to **recipients (1)** and attachment information, message creations, and message deletions are included in **content synchronization** operations.

Hierarchy synchronization and content synchronization operations are the actual processes used to implement ICS on the client and server.

1.3.2.1 Download

Information about all changes and deletions made to mailbox data on the server is downloaded to the client through one or more iterations of a single ROP, whose response buffer can be efficiently packed at the RPC level.

Performing a hierarchy synchronization download operation using a **synchronization download context** that was opened on a folder will produce information about all folder changes and folder deletions of descendants of that folder that have happened since the last synchronization download, as defined by the **initial ICS state**.

Performing a content synchronization download operation using a **synchronization context** that was opened on a folder will produce information about all message changes and message deletions in the folder that have happened since the last synchronization download, as defined by the initial ICS state.

1.3.2.2 Upload

Uploading mailbox changes from a client to a server resembles the ICS download process, except that instead of streaming data through a single ROP, multiple individual ROPs are sent to upload changes to individual objects within a mailbox.

This protocol supports the uploading of hierarchy differences, such as creation and deletion of folders and changes to folder properties.

This specification also supports the uploading of differences in the contents of folders, such as creation and deletion of messages, changes to message properties and read state, and the moving of messages between folders.

1.4 Relationship to Other Protocols

This specification provides a low-level explanation of bulk data transfer operations.

The Mailbox Synchronization Protocol Specification describes how to apply this protocol to the replication of mailbox data between clients and servers. For more information, see [MS-OXCSYNC].

This specification relies on the following:

- An understanding of RPCs and ROPs, as described in [MS-OXCRPC] and [MS-OXCROPS], respectively.
- An understanding of folders and messages, as described in [MS-OXCFOLD] and [MS-OXCMSG], respectively.

1.5 Prerequisites/Preconditions

When performing bulk data transfer operations, this protocol assumes that the client has previously logged on to the server and has acquired a **handle** to the folder that contains the messages and subfolders that will be uploaded or downloaded. For information about folders, see [MS-OXCFOLD].

1.6 Applicability Statement

This protocol was designed for the following uses:

- To support the replication of mailbox content between clients and servers, as further described in [MS-OXCSYNC].
- To support client-driven copying of data between multiple mailboxes on multiple servers.
- To support exporting or importing of data to or from a mailbox.

This protocol provides high efficiency and complete preservation of data fidelity for the uses described in this section. However, use of the protocol is not appropriate in the following scenarios:

- For those copying data between folders in the same mailbox, or different mailboxes residing on the same server. Consider using the **RopCopyTo** ROP, as described in [MS-OXCROPS] section 2.2.8.12, for maximum efficiency.
- For those requiring detailed control over the set of information that has to be transferred for each message. Consider using other ROPs described in [MS-OXCROPS] that provide access to individual parts of messages.
- For those that impose constraints on the amount of data that has to be passed over the wire or stored on the client.
- For those that do not allow for persistence of state information on the client between runs.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Localization:** Localization-related aspects of the protocol are described in section 2.2.3.1.1.1.2.

- **Capability Negotiation:** This protocol performs explicit capability negotiation by using the following ROPs, properties, and **flags**. Support of the following features is determined by the versions of the client and server that are supplied during the connect phase (by the **EcDoConnectEx** RPC) of the RPC session. For more information, see [MS-OXCRPC] section 3.1.9.

Client version	Description
11.0.0.4920 and above	The client supports receiving **ServerBusy** in the **ReturnValue** field of the **RopFastTransferSourceGetBuffer** ROP response. For more information, see section 2.2.3.1.1.5.
12.0.3730.0 and above	The client supports send optimization for ICS using the **PidTagTargetEntryId** property ([MS-OXOMSG] section 2.2.1.66). For more information, see section 3.3.5.5.4.2.1.2.

Server version	Description
8.0.359.0 and above	The server supports **PartialItem SendOptions** flag. For more information, see section 2.2.3.1.1.1.2. Earlier server versions do not support this flag.

The **RopTellVersion** ROP is used to explicitly declare capabilities of the servers in the server-to-client-to-server upload scenario. For more information, see section 3.3.5.4.2.1.

1.8 Vendor-Extensible Fields

This protocol provides no extensibility beyond what is specified in [MS-OXCMSG].

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

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

2.2 Message Syntax

The following notations are used in this specification:

- **PidTagCnset***. Refers to any of the following properties: **PidTagCnsetSeen** (section 2.2.1.1.2), **PidTagCnsetSeenFAI** (section 2.2.1.1.3), and **PidTagCnsetRead** (section 2.2.1.1.4).
- **RopFastTransferSourceCopy***. Refers to any of the following ROPs: **RopFastTransferSourceCopyTo**, **RopFastTransferSourceCopyProperties**, **RopFastTransferSourceCopyMessages**, and **RopFastTransferSourceCopyFolder**.
- **RopSynchronizationImport***. Refers to any of the following ROPs: **RopSynchronizationImportMessageChange**, **RopSynchronizationImportHierarchyChange**, **RopSynchronizationImportMessageMove**, **RopSynchronizationImportDeletes**, and **RopSynchronizationImportReadStateChanges**.

Section 2.2.3.2.4.2 through section 2.2.4.3.25 use **property list restriction tables** in the following format to describe restrictions on arrays of property values:

Name	Restrictions	Comments
PidSomeProperty	Conditional Fixed position ...	When the value of Restrictions column contains "Conditional", the Comments column is specifying the condition of existence. When the value of the Restrictions column does not contain "Conditional", the Comments column can contain comments about the property value.
< other properties >	*Prohibited*	Comments.

Any property cannot exist in a property list restriction table more than once. All nonitalicized rows of the table represent a restriction that is imposed on the property identified in the Name column. For more information about all possible properties, see [MS-OXPROPS]. The Comments column contains free-form comments that amend the meaning of the Name and Restrictions columns. The Restrictions column specifies a subset of the following restrictions:

- Optional [default]: The property can be present in the array.
- Required: The property MUST be present in the array.
- Fixed position: The position of the property within the array is fixed and MUST correspond to the position of the corresponding restriction in the property list restriction table.
- Conditional: The presence of the property in the array is conditional. See the Comments column for conditions.

- Prohibited: The property MUST NOT be present in the array. Italicized rows represent restrictions that apply to special sets of properties. The special set < *other properties* > represents all properties that are not mentioned in the property list restriction table explicitly.
- No restrictions: There are no restrictions on the property in the array. The property does not have a fixed position in the array, and by default, the property's presence is optional in the array.

All undefined bits in flag structures and undefined values of enumerations that are defined in this specification are reserved; clients MUST pass 0. Server behavior for undefined flags and enumeration values is defined in section 3.2.

2.2.1 Properties

2.2.1.1 ICS State Properties

ICS uses a set of properties known as the ICS state to enable a server to narrow down the set of data passed during an Incremental Change Synchronization (ICS). Details about how the ICS state properties are used by the client and server are included in section 3.1.5.2.

All ICS state properties are of the **PtypBinary** type ([MS-OXCDATA] section 2.11.1), and contain a serialized **IDSET** in the **replica GUID (REPLGUID)**-based form, as specified in section 2.2.2.4.1.

All properties specified in this section are part of the ICS state. Two of these properties are used for hierarchy synchronization operations: **PidTagIdsetGiven** (section 2.2.1.1.1) and **PidTagCnsetSeen** (section 2.2.1.1.2). All four properties are used for content synchronization operations.

The ICS state determines the state of the **local replica** bounded by the **synchronization scope** specified by the client in the **RopSynchronizationConfigure** ROP request (section 2.2.3.2.1.1).

2.2.1.1.1 PidTagIdsetGiven

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1), that contains a serialization of REPLGUID-based **IDSETs**. The **IDSETs** contain **FID** values ([MS-OXCDATA] section 2.2.1.1) for hierarchy synchronization operations, or **MID** values for content synchronization operations, that exist in the local replica of the client. This **IDSET** MUST NOT include any IDs that are not in the local replica of the client. Because of this restriction on IDs, this property might not compress as well as the **PidTagCnset*** properties, which will make the **PidTagIdsetGiven** property grow much larger than the **PidTagCnset*** properties. For more details about compression of **IDSETs**, see section 3.1.5.6.

For more details about sending and receiving this property, see section 3.1.5.2.1.

2.2.1.1.2 PidTagCnsetSeen

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of REPLGUID-based **CNSET**s, as specified in section 2.2.2.4. The **CN** structures, as specified in section 2.2.2.1, in the **CNSET** track changes to folders (for hierarchy synchronization operations) or **normal messages** (for content synchronization operations) in the current synchronization scope that have been previously communicated to a client, and are reflected in its local replica.

2.2.1.1.3 PidTagCnsetSeenFAI

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of REPLGUID-based **IDSETs,** with semantics identical to the **PidTagCnsetSeen** property (section 2.2.1.1.2),

except that this property contains IDs for **folder associated information (FAI)** messages and is therefore only used in content synchronization operations.

2.2.1.1.4 PidTagCnsetRead

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of REPLGUID-based **CNSETs**. The **CN** structures, as specified in section 2.2.2.1, in the **CNSET** track changes to the read state for messages in the current synchronization scope that have been previously communicated to the client and are reflected in its local replica.

The read state of a message is determined by the **PidTagMessageFlags** property ([MS-OXCMSG] section 2.2.1.6), which contains a bitmask of flags that indicate the origin and current state of the message.

2.2.1.2 Messaging Object Identification and Change Tracking Properties

This section specifies details about the properties that are used by this protocol to identify messages, folders, and track changes.

For details about how messaging object and change identification values are created and modified by the protocol roles, see section 3.1.5.3.

2.2.1.2.1 PidTagMid

A **PtypInteger64** value ([MS-OXCDATA] section 2.11.1) that contains the **MID** value ([MS-OXCDATA] section 2.2.1.2) of the message currently being synchronized.

For details about the conditions of its presence in message change headers, see the **SynchronizationExtraFlag** field in section 2.2.3.2.1.1.1.

2.2.1.2.2 PidTagFolderId

A **PtypInteger64** value ([MS-OXCDATA] section 2.11.1) that contains the **FID** value ([MS-OXCDATA] section 2.2.1.1) of the folder currently being synchronized.

For details about the conditions of its presence in message change headers, see the **SynchronizationExtraFlag** field in section 2.2.3.2.1.1.1.

2.2.1.2.3 PidTagChangeNumber

A **PtypInteger64** value ([MS-OXCDATA] section 2.11.1) that contains the **CN** structure, as specified in section 2.2.2.1, that identifies the last change to the message or folder that is currently being synchronized.

For details about the conditions of its presence in message change headers, see the **SynchronizationExtraFlag** field in section 2.2.3.2.1.1.1.

2.2.1.2.4 PidTagParentFolderId

A **PtypInteger64** value ([MS-OXCDATA] section 2.11.1) that contains the **FID** value ([MS-OXCDATA] section 2.2.1.1) that identifies the parent folder of the messaging object being synchronized.

2.2.1.2.5 PidTagSourceKey

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains an **internal identifier (2)** for this folder or message. The binary content of this property is a serialization of an **XID**. For more details about the binary format, see section 2.2.2.2.

For more details about how clients generate this property, see section 3.3.5.1.1. For more details about how servers output this property value or generate it on-the-fly, see section 3.2.5.2.

2.2.1.2.6 PidTagParentSourceKey

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) on a folder, that contains the **PidTagSourceKey** property (section 2.2.1.2.5) of the folder's parent folder.

2.2.1.2.7 PidTagChangeKey

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains the serialized **XID**, as specified in section 2.2.2.2, of the last change to the messaging object.

If the last change to the messaging object was imported from a client by using the **RopSynchronizationImportMessageChange** ROP, this property contains a value for the **PidTagChangeKey** property that was passed in fields to that ROP.

If the last change to a messaging object was made by a server, this property contains an **XID**, as specified in section 2.2.2.2, generated from the **PidTagChangeNumber** property (section 2.2.1.2.3). For more details about generating **XIDs** based on internal identifiers (2), see section 3.2.5.2.

2.2.1.2.8 PidTagPredecessorChangeList

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialized representation of a **PredecessorChangeList** structure, as specified in section 2.2.2.3. This value represents a set of **CN** structures, as specified in section 2.2.2.1, for versions of the messaging object in all **replicas (2)** that were integrated into the current version. This property is used in **conflict detection** by all protocol roles.

2.2.1.3 Properties for Encoding Differences in Replica Content

Because servers do not maintain a per-client state, the following properties are not persisted on servers and are only present as data in the FastTransfer streams.

All properties are of the **PtypBinary** ([MS-OXCDATA] section 2.11.1) type, and contain a serialized **IDSET** in the **REPLID**-based form, as specified in section 2.2.2.4.1.

2.2.1.3.1 PidTagIdsetDeleted

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of a REPLID-based **IDSETs**. The **IDSETs** contain the IDs of folders (for hierarchy synchronization operations) or messages (for content synchronization operations) that were **hard deleted** or **soft deleted** since the last synchronization identified by the initial ICS state. For more details about how an **IDSET** is serialized, see section 3.1.5.5.

2.2.1.3.2 PidTagIdsetNoLongerInScope

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of a REPLID-based **IDSETs**. The **IDSETs** contain the IDs of messages that got out of the synchronization scope since the last synchronization identified by the initial ICS state. Messages that no longer match a restriction are considered out of synchronization scope. For more details about how an **IDSET** is serialized, see section 3.1.5.5.

Note that messages moved to another folder are considered soft deleted in the source folder; hard deleted and soft deleted messages are reported in the **PidTagIdsetDeleted** property (section 2.2.1.3.1).

2.2.1.3.3 PidTagIdsetExpired

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of REPLID-based **IDSETs**. The **IDSETs** contain IDs of **expired Message objects** in a **public folder** that expired since the last synchronization identified by the initial ICS state. For more details about how an **IDSET** is serialized, see section 3.1.5.5.

2.2.1.3.4 PidTagIdsetRead

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of REPLID-based **IDSETs**. The **IDSETs** contain IDs of messages that were marked as read (as specified by the **PidTagMessageStatus** property in [MS-OXCMSG] section 2.2.1.8) since the last synchronization identified by the initial ICS state. For more details about how an **IDSET** is serialized, see section 3.1.5.5.

2.2.1.3.5 PidTagIdsetUnread

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that contains a serialization of REPLID-based **IDSETs**. The **IDSETs** contain IDs of messages that were marked as unread (as specified by the **PidTagMessageStatus** property in [MS-OXCMSG] section 2.2.1.8) since the last synchronization identified by the initial ICS state. For more details about how an **IDSET** is serialized, see section 3.1.5.5.

2.2.1.4 PidTagAssociated

A **PtypBoolean** value ([MS-OXCDATA] section 2.11.1) that specifies whether the message being synchronized is an FAI message.

2.2.1.5 PidTagMessageSize

An unsigned **PtypInteger32** value ([MS-OXCDATA] section 2.11.1) that identifies the size of the message in bytes.

For details about the **conditions** of the **PidTagMessageSize** presence in message change headers, see section 2.2.3.2.1.1.1.

2.2.1.6 PidTagResolveMethod

A **PtypInteger32** value ([MS-OXCDATA] section 2.11.1) that specifies how to resolve any conflicts with the message. This property is not required. However, if it is set, flags other than the following MUST NOT be present:

- RESOLVE_METHOD_DEFAULT (0x00000000). A conflict resolve message SHOULD be generated.

- RESOLVE_METHOD_LAST_WRITER_WINS (0x00000001). Overwrite the target message with current changes being applied.
- RESOLVE_NO_CONFLICT_NOTIFICATION (0x00000002). Do not send a conflict notification message when generating a conflict resolve message in a public folder.

A client or server MUST NOT generate conflict resolve messages for FAI messages. These messages MUST be resolved by using RESOLVE_METHOD_LAST_WRITER_WINS semantics, as specified in section 3.1.4.1.2.2.

For more details about **conflict resolution**, see section 3.1.4.1.

2.2.1.7 Properties That Denote Subobjects

The properties in the following tables denote subobjects of the messaging objects and can be used in the following:

- The property inclusion and exclusion lists of ROPs that configure download operations. For example, the **RopSynchronizationConfigure** and **RopFastTransferSourceCopyTo** ROPs.
- As values of **PidTagFXDelProp** meta-properties, as specified in section 2.2.4.1.5.1.

Folder Properties	Description
PidTagContainerContents ([MS-OXPROPS] section 2.713)	Identifies all normal messages in the current folder.
PidTagFolderAssociatedContents ([MS-OXPROPS] section 2.775)	Identifies all FAI messages in the current folder.
PidTagContainerHierarchy ([MS-OXPROPS] section 2.715)	Identifies all subfolders of the current folder.<1>

Message Properties	Description
PidTagMessageRecipients ([MS-OXPROPS] section 2.892)	Identifies all recipients (1) of the current message.
PidTagMessageAttachments ([MS-OXPROPS] section 2.882)	Identifies all attachments to the current message.

Attachment Properties	Description
PidTagAttachDataObject ([MS-OXCMSG] section 2.2.2.8)	Identifies the **Embedded Message object** of the current attachment.<2>

2.2.2 Structures

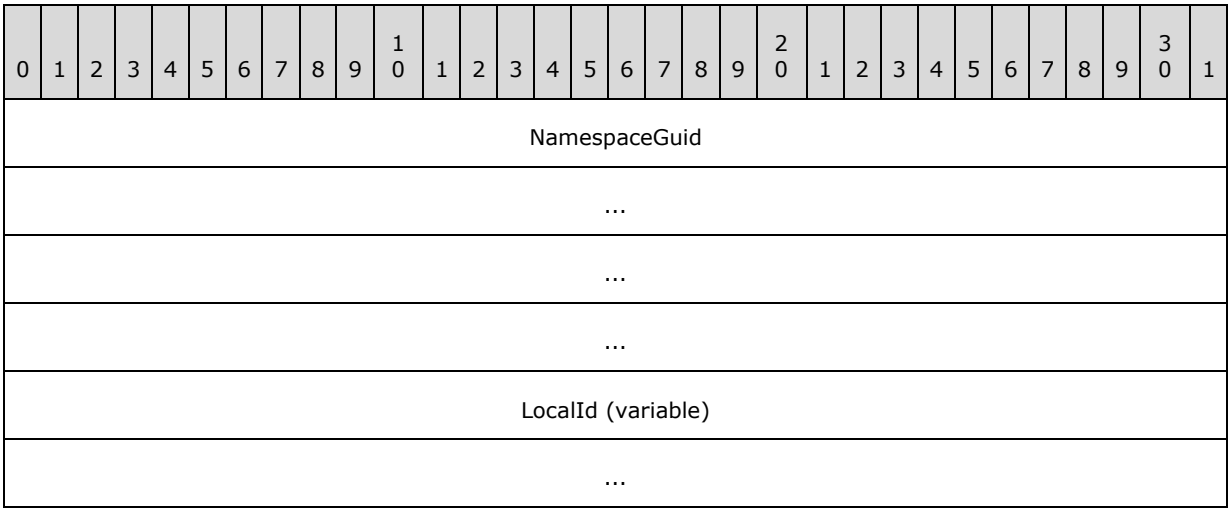
2.2.2.1 CN

A **CN** structure represents a **change number** that identifies a version of a messaging object. Change numbers are identical in format to **FID** structures ([MS-OXCDATA] section 2.2.1.1) and

MID structures ([MS-OXCDATA] section 2.2.1.2), except the **GlobalCounter** field represents a change to a messaging object rather than a messaging object itself.

2.2.2.2 XID

An **XID** structure represents an **external identifier** for an entity within a data **store**.



NamespaceGuid (16 bytes): A 128-bit **GUID** that identifies the namespace that the identifier specified by **LocalId** belongs to.

LocalId (variable): A variable binary value that contains the ID of the entity in the namespace specified by **NamespaceGuid**. This field has a minimum length of 1 byte and a maximum length of 8 bytes.

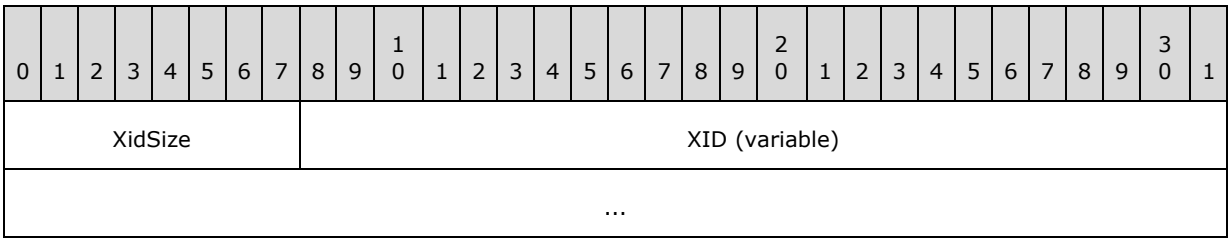
For more details about **GID** structures, which are a subtype of an **XID**, see [MS-OXCDATA] section 2.2.1.3. For **GIDs**, the REPLGUID maps to the **NamespaceGuid** field, and the **GlobalCounter** maps to the **LocalId** field.

All **XIDs** with the same **NamespaceGuid** MUST have the same length of **LocalId** fields. However, the size of the **LocalId** value cannot be determined by examining the **NamespaceGuid** value and MUST be provided externally. In most cases, **XID** structures are present within other structures, which specify the size of the **XID**, such as the **SizedXid** element, as specified in section 2.2.2.3.1 or the **propValue** element, as specified in section 2.2.4.3.21.

2.2.2.3 PredecessorChangeList

The **PredecessorChangeList** structure contains a set of **XIDs**, as specified in section 2.2.2.2, that represent change numbers of messaging objects in different replicas (2). The order of the **XIDs** does not have significance for interpretation, but is significant for serialization and deserialization. The set of **XIDs** MUST be serialized without padding as an array of **SizedXid** structures binary-sorted by the value of **NamespaceGuid** field of the **XID** structure in the ascending order.

2.2.2.3.1 SizedXid



XidSize (1 byte): An unsigned 8-bit integer. MUST be equal to the size of the XID field in bytes.

XID (variable): A structure of type **XID**, as specified in section 2.2.2.2, that contains the value of the internal identifier (2) of an object, or internal or external identifier of a change number. This field MUST contain the same number of bytes as specified in the **XidSize** field.

2.2.2.4 IDSET and CNSET

An **IDSET** structure contains a set of ID values. The ID values are one of the following types:

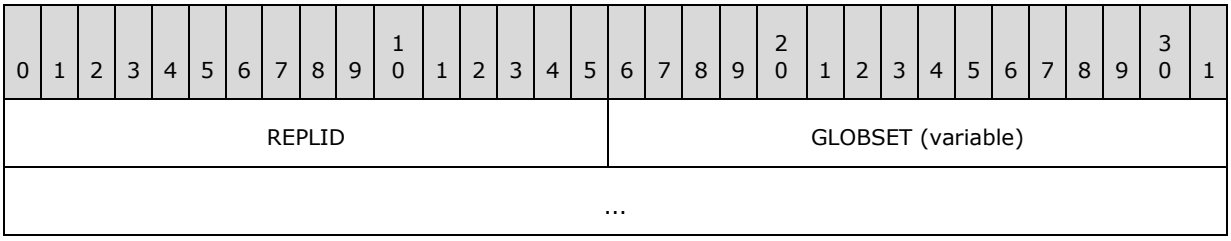
- **MID** structures ([MS-OXCDATA] section 2.2.1.2)
- **FID** structures ([MS-OXCDATA] section 2.2.1.1)
- **CN** structures, as specified in section 2.2.2.1.

When an **IDSET** structure contains **CN** structures, it is also known as a **CNSET**. In this section, the term **IDSET** is used to refer to both **IDSETs** and **CNSETs**.

The **IDSET** serialization format specified in the following sections is optimized for data transfer, and is not intended for in-memory operations. For details about the serialization and deserialization process, see section 3.1.5.5.

2.2.2.4.1 Serialized IDSET with REPLID

For every REPLID and **GLOBSET** pair represented in the formatted **IDSET**, add the following to the serialization buffer in lowest to highest REPLID order.

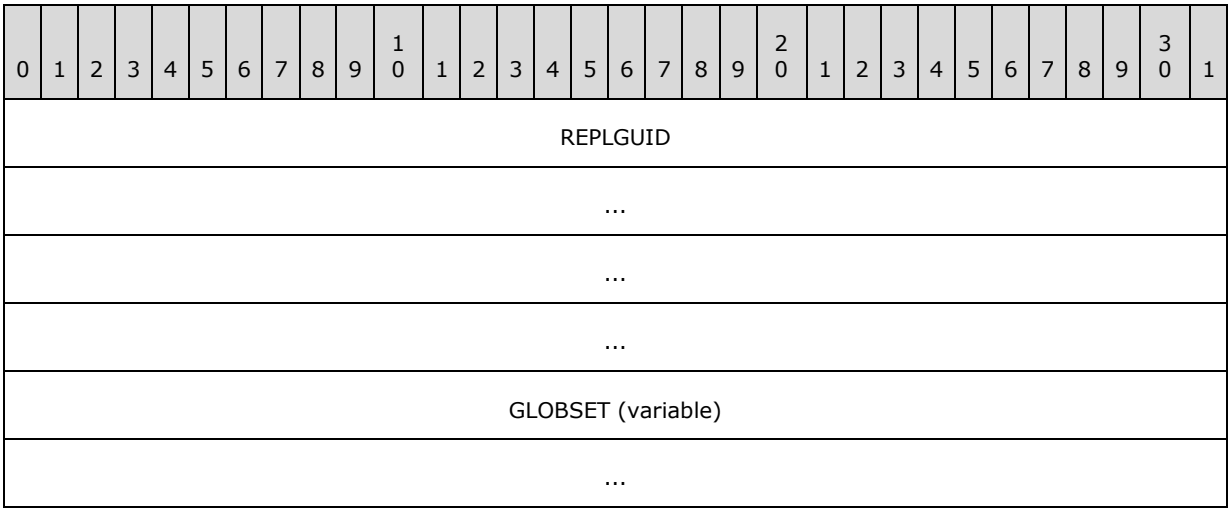


REPLID (2 bytes): A REPLID value that when combined with all GLOBCNT values represented in the GLOBSET field, produces a set of IDs.

GLOBSET (variable): A serialized **GLOBSET**.

2.2.2.4.2 Serialized IDSET with REPLGUID

For every REPLGUID and **GLOBSET** pair represented in the formatted **IDSET**, add the following to the serialization buffer. REPLGUID-**GLOBSET** pairs MUST be serialized by REPLGUID in the ascending order, using byte-to-byte comparison.



REPLGUID (16 bytes): A GUID value that represents a REPLGUID. When combined with all GLOBCNT values represented in the GLOBSET field, produces a set of **GID** values ([MS-OXCDATA] section 2.2.1.3). The GUID values can be converted into a REPLID to produce a set of IDs.

GLOBSET (variable): A serialized **GLOBSET**.

2.2.2.5 GLOBSET

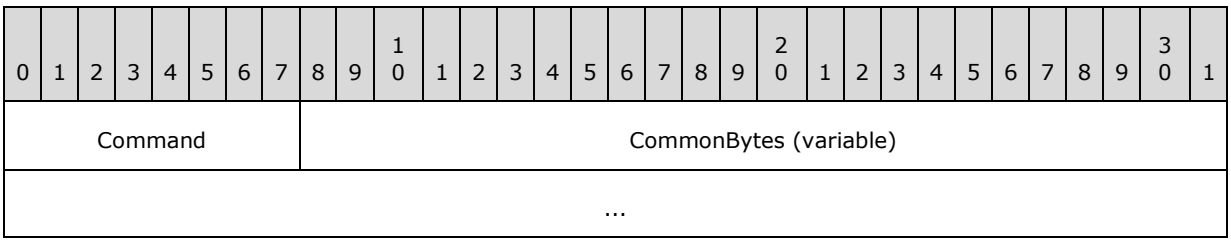
A **GLOBSET** is a set of GLOBCNT values that are reduced to one or more GLOBCNT ranges. A single GLOBCNT range identifies only the lowest and highest values in a set of consecutive GLOBCNT values. A GLOBCNT range is created using any of the commands in this section, with the exception of the **Pop** and **End** commands.

The serialization format specified in the following sections is optimized for data transfer, and is not intended for in-memory operations.

A **GLOBSET** is serialized without padding as a set of commands. For details about how to translate an abstract data model for a **GLOBSET** into a set of commands, see section 3.1.5.5.3.

2.2.2.5.1 Push Command (0x01 – 0x06)

The **Push** command will place high-order bytes onto the **common byte stack**.

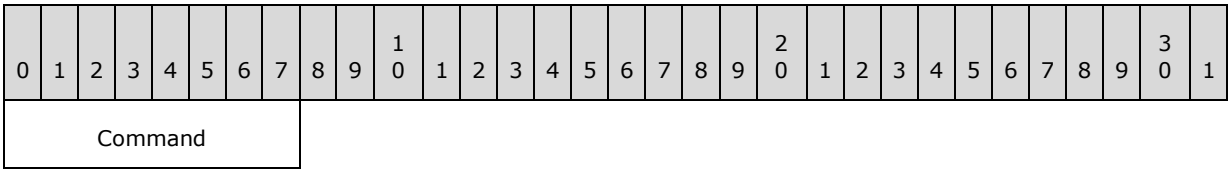


Command (1 byte): A value in the range "0x01" through "0x06".

CommonBytes (variable): Variable length byte array to be pushed onto the common byte stack. The length of the byte array is equal to the **Command** value ("0x01" through "0x06").

2.2.2.5.2 Pop Command (0x50)

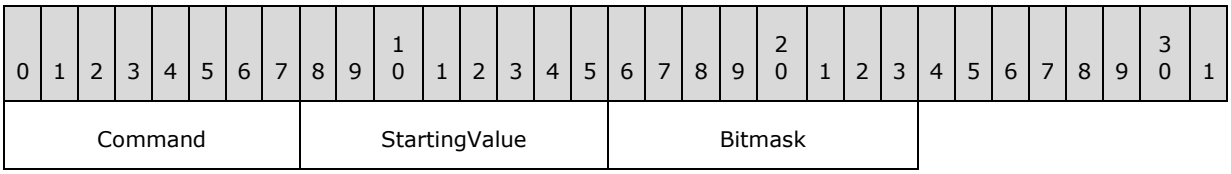
The **Pop** command will remove bytes that were added to the common byte stack from the previous **Push** command.



Command (1 byte): The value "0x50".

2.2.2.5.3 Bitmask Command (0x42)

The **Bitmask** command allows for up to five GLOBCNT ranges to be compressed into a single encoding command if they all have five high-order bytes in common and the low-order bytes are all within eight values of each other.



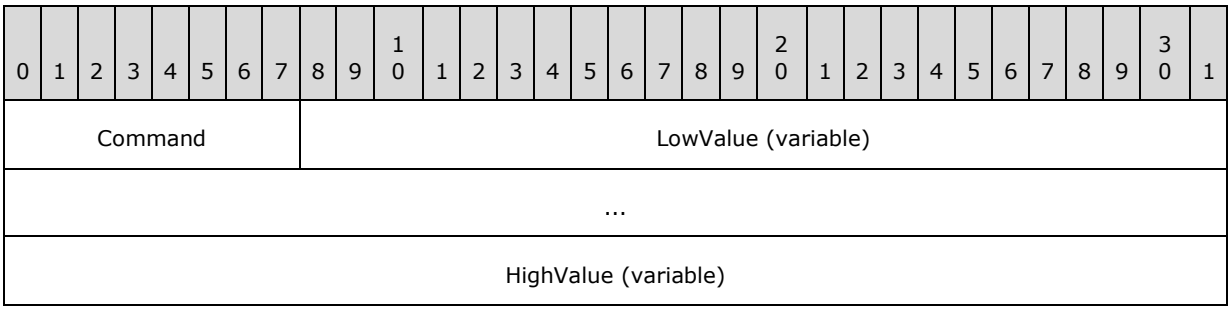
Command (1 byte): The value "0x42".

StartingValue (1 byte): Low-order byte of first GLOBCNT.

Bitmask (1 byte): Used for GLOBCNT generation where values are defined based on the **StartingValue** and which bits are set in **Bitmask**.

2.2.2.5.4 Range Command (0x52)

The **Range** command is used to add a GLOBCNT range to the **GLOBSET**. The range is determined by the GLOBCNT value produced from the **LowValue** field and the GLOBCNT produced from the **HighValue** field.



...

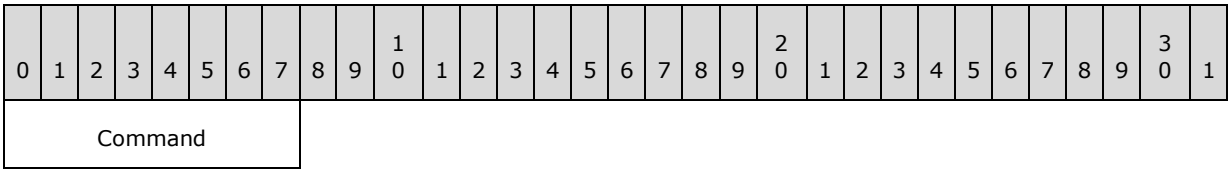
Command (1 byte): The value "0x52".

LowValue (variable): Variable length byte array of low-order values for GLOBCNT generation. The number of bytes in this field is equal to six minus the number of high-order bytes in the common byte stack. MUST be less than or equal to **HighValue**.

HighValue (variable): Variable length byte array of low-order values for GLOBCNT generation. The number of bytes in this field is equal to six minus the number of high-order bytes in the common byte stack. MUST be greater or equal to **LowValue**.

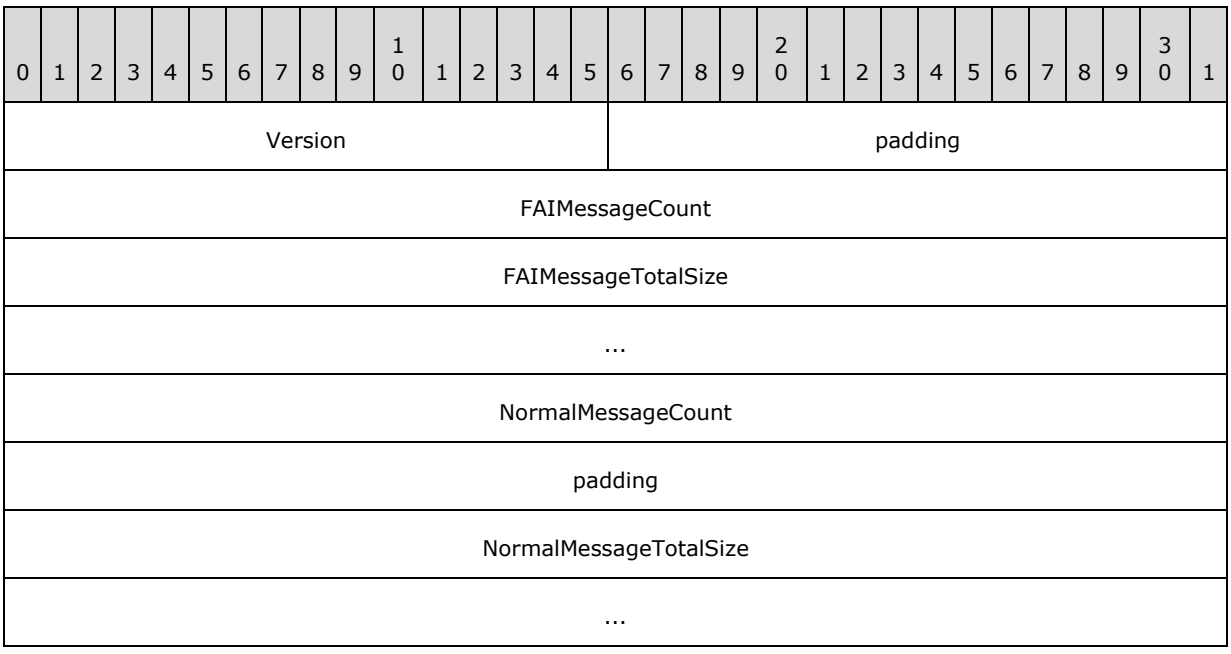
2.2.2.5.5 End Command (0x00)

The **End** command is used to signal the end of the **GLOBSET** encoding.



Command (1 byte): The value "0x00".

2.2.2.6 ProgressInformation



Version (2 bytes): An unsigned 16-bit value that contains a number that identifies the binary structure of the data that follows. The table in this section describes a format for version "0x0000", which is the only version of this structure defined for this protocol.

padding (2 bytes): SHOULD be set to zeros and MUST be ignored by clients.

FAIMessageCount (4 bytes): An unsigned 32-bit integer value that contains the total number of changes to FAI messages that are scheduled for download during the current synchronization operation.

FAIMessageTotalSize (8 bytes): An unsigned 64-bit integer value that contains the size in bytes of all changes to FAI messages that are scheduled for download during the current synchronization operation.

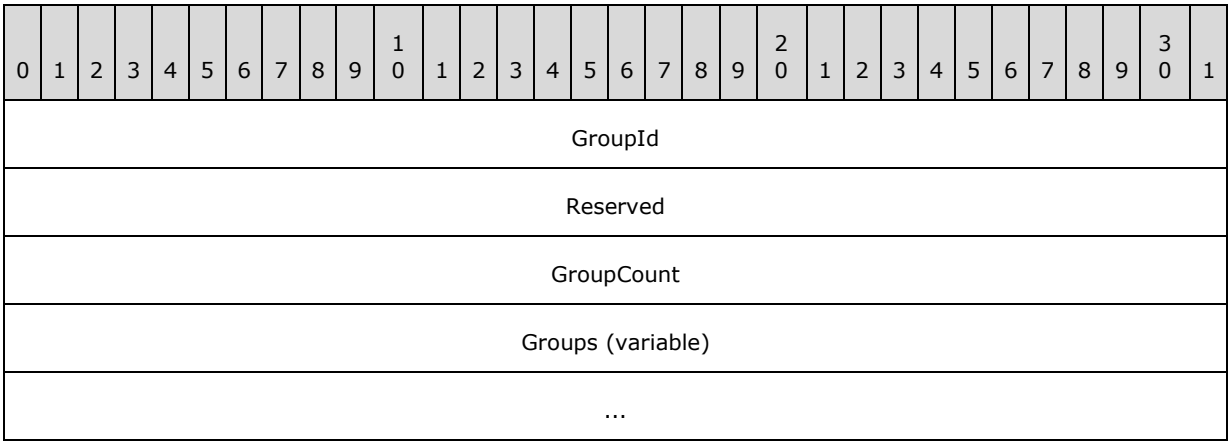
NormalMessageCount (4 bytes): An unsigned 32-bit integer value that contains the total number of changes to normal messages that are scheduled for download during the current synchronization operation.

padding (4 bytes): SHOULD be set to zeros and MUST be ignored by clients.

NormalMessageTotalSize (8 bytes): An unsigned 64-bit integer value that contains the size in bytes of all changes to normal messages that are scheduled for download during the current synchronization operation.

2.2.2.7 PropertyGroupInfo

The **PropertyGroupInfo** structure describes a single property mapping between a group index and **property tags** within a property group. For more details about property groups, see section 3.1.5.4.



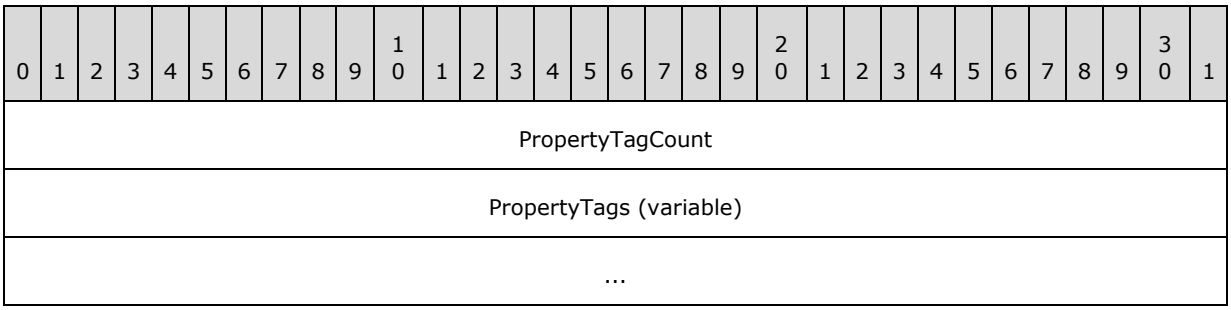
GroupId (4 bytes): An unsigned 32-bit integer value that identifies a property mapping within the current synchronization download.

Reserved (4 bytes): This value MUST be set to "0x00000000".

GroupCount (4 bytes): An unsigned 32-bit integer value that specifies how many **PropertyGroup** structures are present in the **Groups** field. MUST NOT be zero ("0x00000000").

Groups (variable): An array of **PropertyGroup** structures. This field MUST contain **GroupCount** **PropertyGroup** elements.

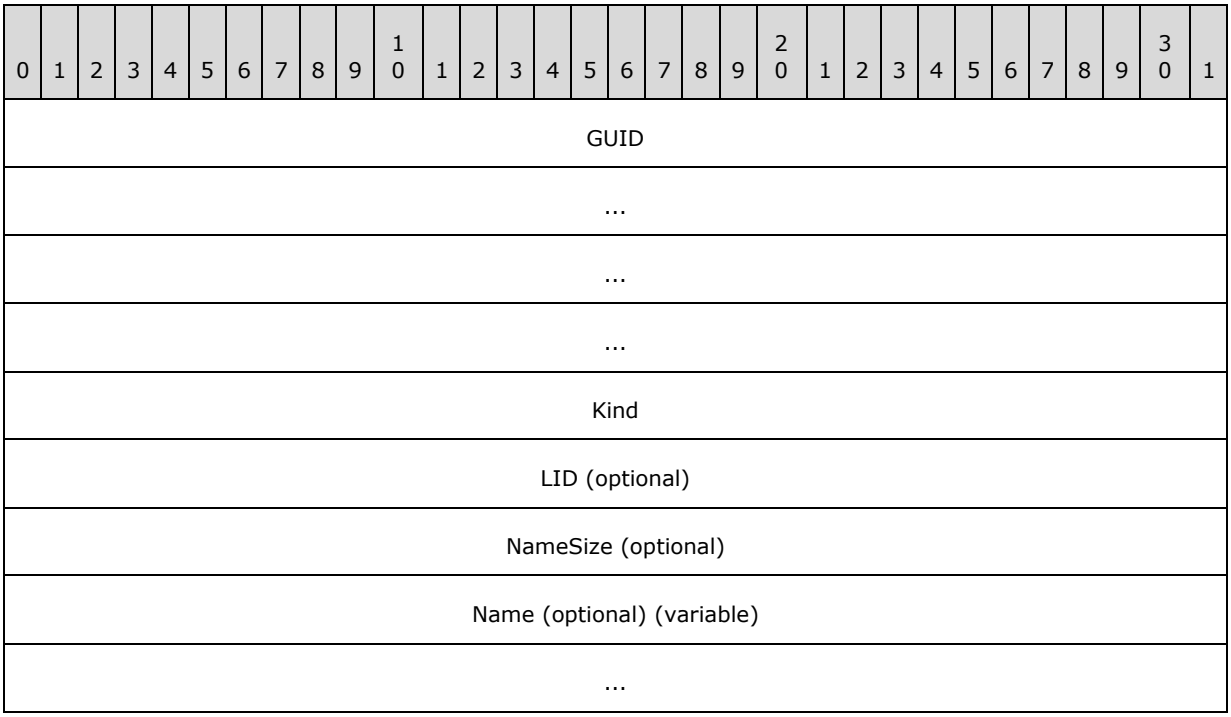
2.2.2.7.1 PropertyGroup



PropertyTagCount (4 bytes): An unsigned 32-bit integer value that specifies how many **PropertyTag** structures are present in **PropertyTags**. MUST NOT be zero ("0x00000000").

PropertyTags (variable): A variable length array of **PropertyTag** structures ([MS-OXCDATA] section 2.9). If a **PropertyTag** identifies a **named property**, the **PropertyTag** is immediately followed by a **GroupPropertyName** structure, as specified in section 2.2.2.7.1.1. Named properties are identified by a **PropertyId** value ([MS-OXCDATA] section 2.9) greater than or equal to 0x8000. This field MUST contain **PropertyTagCount** tags.

2.2.2.7.1.1 GroupPropertyName



GUID (16 bytes): The GUID that identifies the **property set** for the named property.

Kind (4 bytes): The following are possible values for the **Kind** field:

Name	Value
0x00000000	The property is identified by the LID field.
0x00000001	The property is identified by the Name field.

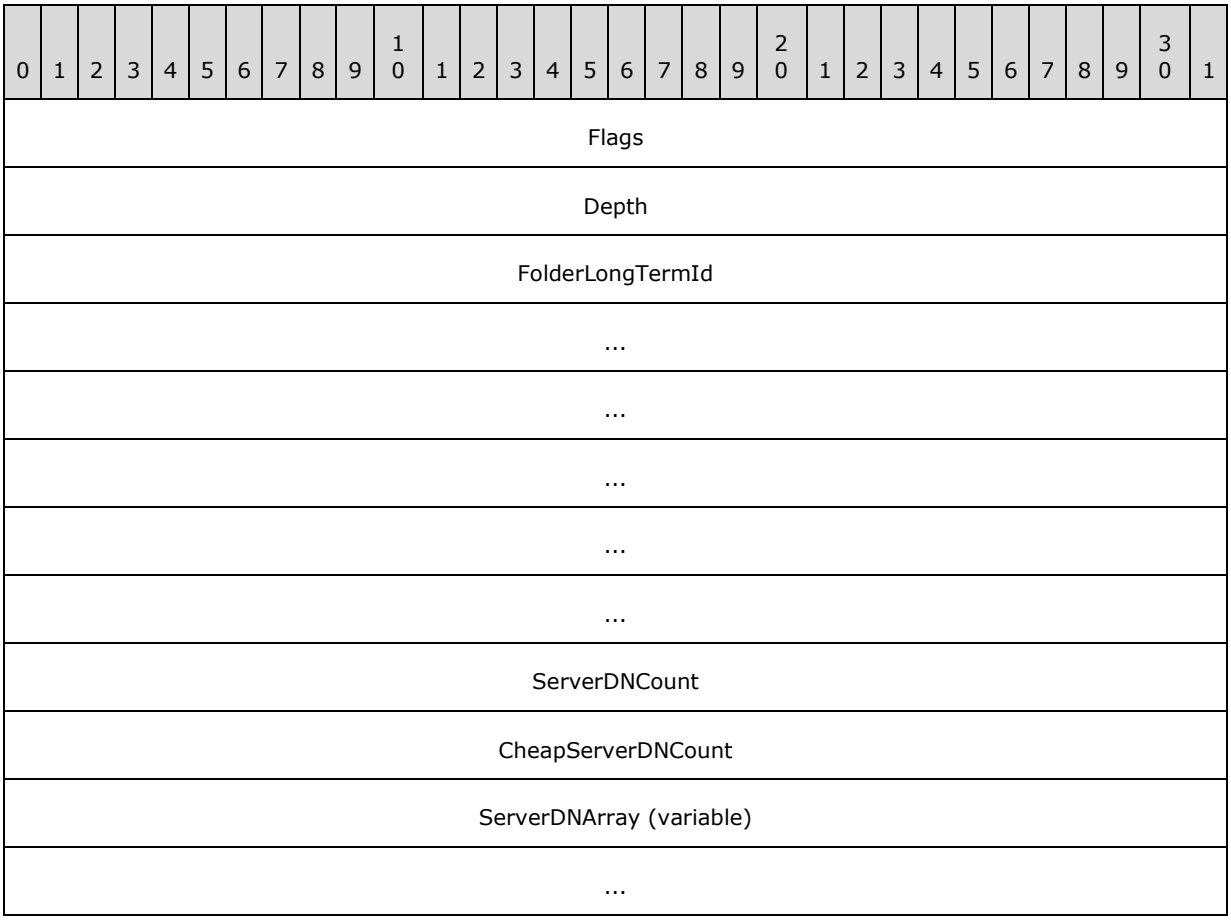
LID (optional) (4 bytes): Present only if **Kind** is set to "0x00000000". An unsigned integer that identifies the named property within its property set.

NameSize (optional) (4 bytes): Present only if **Kind** is set to "0x00000001". Identifies the number of bytes in the **Name** string.

Name (optional) (variable): Present only if **Kind** is set to "0x00000001". A **Unicode** (UTF-16) string, followed by two zero bytes as a null terminator, that identifies the property within its property set.

2.2.2.8 FolderReplicaInfo

The **FolderReplicaInfo** structure contains information about **server replicas** of a public folder.



Flags (4 bytes): MUST be set to "0x00000000".

Depth (4 bytes): MUST be set to "0x00000000".

FolderLongTermId (24 bytes): A **LongTermID** structure. Contains the LongTermID of a folder, for which server replica information is being described.

ServerDNCount (4 bytes): An unsigned 32-bit integer value that determines how many elements exist in **ServerDNArray**. MUST NOT be zero ("0x00000000").

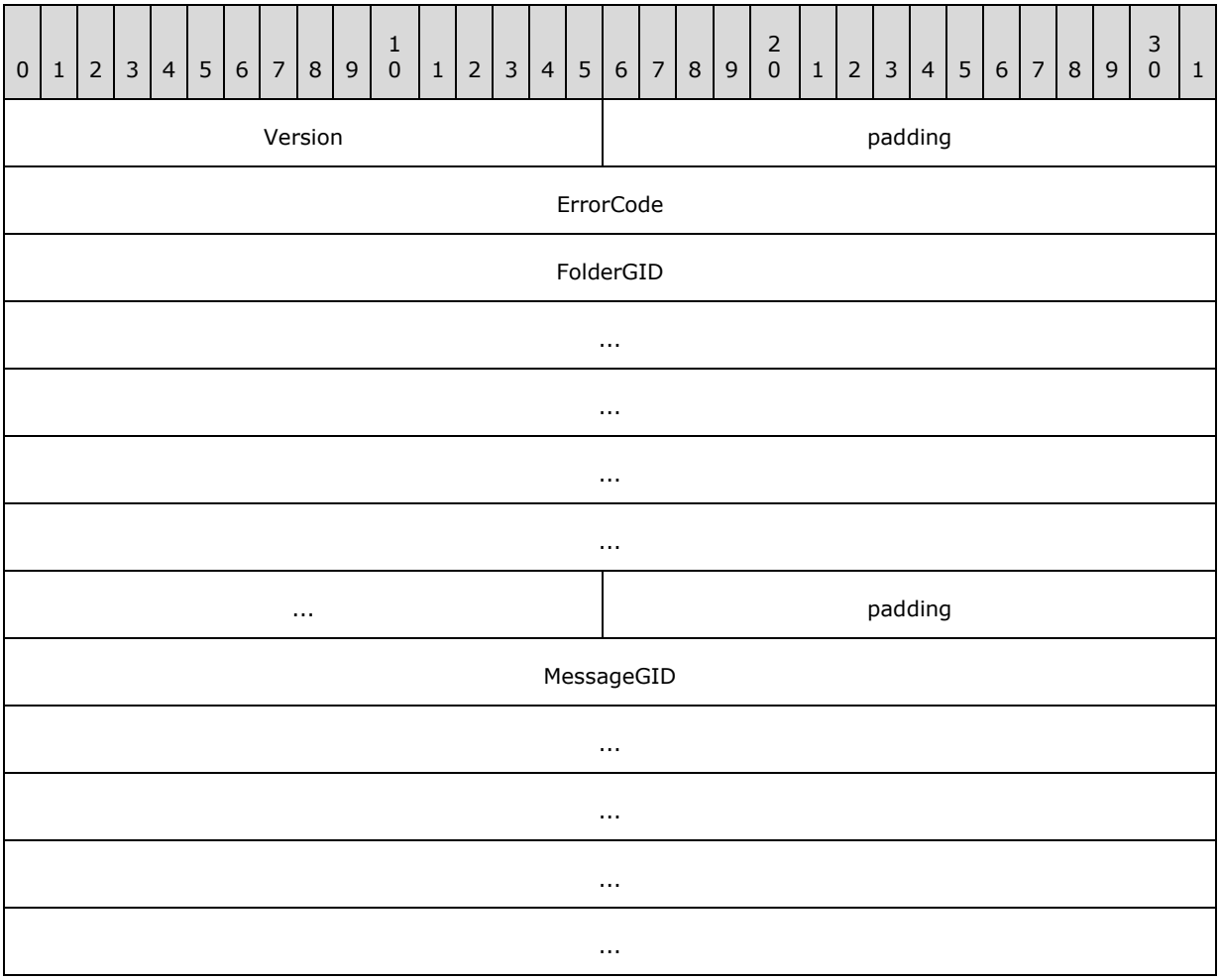
CheapServerDNCount (4 bytes): An unsigned 32-bit integer value that determines how many of the leading elements in **ServerDNArray** have the same, lowest, network access cost. **CheapServerDNCount** MUST be less than or equal to **ServerDNCount**.

ServerDNArray (variable): An array of **ASCII**-encoded NULL-terminated strings. MUST contain **ServerDNCount** strings. Contains an **enterprise/site/server distinguished name (ESSDN)** of servers that have a replica of the folder identifier by **FolderLongTermId**.

2.2.2.9 ExtendedErrorInfo

The **ExtendedErrorInfo** structure contains extended and contextual information about an error that has occurred when producing a FastTransfer stream.

For details about how this structure is used in FastTransfer error recovery and reporting of **partial completion** of download operations, see section 2.2.4.3.4.



...	padding
Reserved	
...	
...	
...	
...	
...	
AuxBytesCount	
AuxBytesOffset	
Reserved (variable)	
...	
AuxBytes (variable)	
...	

Version (2 bytes): An unsigned 16-bit integer that determines the format of the structure. The format shown in the preceding packet diagram corresponds to version "0x00000000", which is the only version defined for the protocol.

padding (2 bytes): SHOULD be set to zeros and MUST be ignored by the clients.

ErrorCode (4 bytes): One of the error codes in the Data Structures Protocol, as specified in [MS-OXCDATA], that describes the reason for the failure.

FolderGID (22 bytes): A **GID** structure ([MS-OXCDATA] section 2.2.1.3) that identifies the folder that was in context at the time the error occurred. MUST be filled with zeros, if no folders were in context.

padding (2 bytes): SHOULD be set to zeros and MUST be ignored by the clients.

MessageGID (22 bytes): A **GID** structure that identifies the message that was in context at the time the error occurred. MUST be filled with zeros, if no messages were in context.

padding (2 bytes): SHOULD be set to zeros and MUST be ignored by the clients.

Reserved (24 bytes): SHOULD be set to zeros and SHOULD be ignored by clients.

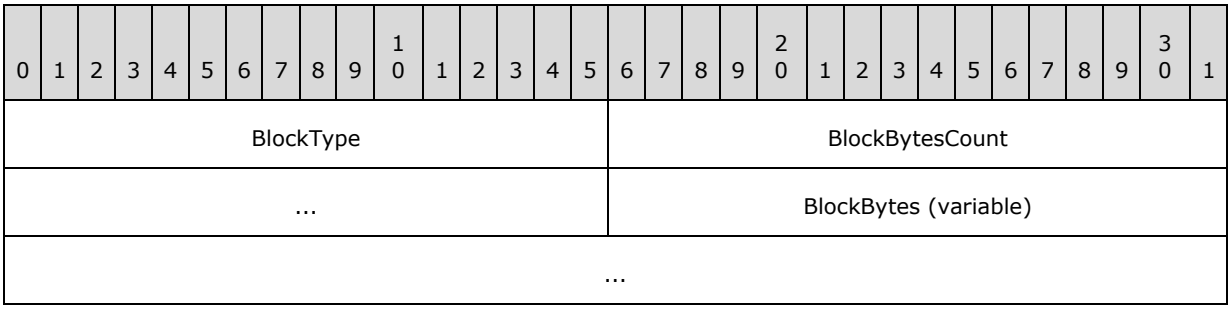
AuxBytesCount (4 bytes): An unsigned 32-bit integer value that specifies the size of the **AuxBytes** field. If set to 0, **AuxBytes** is missing.

AuxBytesOffset (4 bytes): An unsigned 32-bit integer value that specifies the offset in bytes of **Auxbytes** from the beginning of the structure.

Reserved (variable): Optional. SHOULD be set to zeros and SHOULD be ignored by clients.

AuxBytes (variable): An optional **PtypBinary** ([MS-OXCDATA] section 2.11.1) value that MUST be present and reside at **AuxBytesOffset** from the beginning of the structure, if and only if **AuxBytesCount** > 0. If present, MUST consist of one or more **AuxBlock** structures serialized sequentially without any padding.

2.2.2.9.1 AuxBlock



BlockType (2 bytes): An unsigned 16-bit integer that specifies the format of the **BlockBytes** field.

BlockBytesCount (4 bytes): An unsigned 32-bit integer value that specifies the size in bytes of the **BlockBytes** field. The value of **BlockBytesCount** is zero ("0x00000000") if **BlockBytes** contains no data.

BlockBytes (variable): A **PtypBinary** ([MS-OXCDATA] section 2.11.1) value. Semantics are determined by the value of the **BlockType** field. MUST be exactly **BlockBytesCount** bytes long.

Clients MUST ignore any **AuxBlock** structures whose **BlockType** they do not recognize. Unknown **AuxBlocks** can be easily skipped over to subsequent blocks, because their size can always be determined based on **BlockBytesCount**.

2.2.3 ROPs

FastTransfer and ICS operations are performed by sending a specific set of ROP requests to the server.

If a ROP name starts with RopSynchronization, it can only be used in ICS operations.

If a ROP name starts with RopFastTransfer, it can be used in FastTransfer operations, and can also be used ICS operations. See the ROP details provided in this section and the following table for more details.

All FastTransfer and ICS operations can be separated into similar steps:

1. Initialization: Configure an operation and assign it a context, which is used to identify this operation in all subsequent steps.

2. Data transmission: Transmit the messaging object data based on the context configuration.

3. Checkpointing: An optional step in which data that is required for subsequent initialization of the next iteration of this operation is downloaded. For more details about checkpointing, see section 3.3.5.3.

4. Release of resources held on a server: Release the context by using the **RopRelease** ROP. For more details, see [MS-OXCROPS] section 2.2.15.3.

Note that the context in step 1 is not a messaging object, which means that it is not persisted in a mailbox and its lifetime is limited to the lifetime of the handle that is opened for it.

The following table describes the applicability of ROPs for each step of every FastTransfer or ICS operation. See the ROP details in this section for usage directions.

Operation	Initialization	Data transmission	Checkpointing
FastTransfer download	RopFastTransfer **SourceCopy*** ([MS-OXCROPS] section 2.2.12) **RopTellVersion** ([MS-OXCROPS] section 2.2.12.8)	RopFastTransfer **SourceGetBuffer** ([MS-OXCROPS] section 2.2.12.3) Mailbox data is encoded into a FastTransfer stream.	Not applicable.
FastTransfer upload	RopFastTransfer **DestinationConfigure** ([MS-OXCROPS] section 2.2.12.1) **RopTellVersion** ([MS-OXCROPS] section 2.2.12.8)	RopFastTransfer **DestinationPutBuffer** ([MS-OXCROPS] section 2.2.12.2) Mailbox data is encoded into a FastTransfer stream.	Not applicable.
ICS download	RopSynchronization **Configure** ([MS-OXCROPS] section 2.2.13.1) **UploadStateStream*** ([MS-OXCROPS] section 2.2.13)	RopFastTransfer **SourceGetBuffer** ([MS-OXCROPS] section 2.2.12.3) Mailbox data is encoded into a FastTransfer stream.	Not applicable.<3>
ICS upload	RopSynchronization **OpenCollector** ([MS-OXCROPS] section 2.2.13.7) **UploadStateStream*** ([MS-OXCROPS] section 2.2.13)	RopSynchronization **Import*** ([MS-OXCROPS] section 2.2.13) ROPs that operate on a Message object.	RopSynchronization **GetTransferState** ([MS-OXCROPS] section 2.2.13.8) RopFastTransfer **SourceGetBuffer** ([MS-OXCROPS] section 2.2.12.3)

In this section, whenever the applicability of a ROP or protocol details are discussed, operations to which an explanation applies will usually be referenced by mentioning the type of the context, as specified in the following table.

Context type	Operations it applies to
Download context	FastTransfer download, ICS download
FastTransfer context	FastTransfer download, FastTransfer upload
FastTransfer download context	FastTransfer download
FastTransfer upload context	FastTransfer upload
Synchronization context	ICS download, ICS upload

Context type	Operations it applies to
Synchronization download context	ICS download
Synchronization upload context	ICS upload

For details about the relationship between the **InputHandleIndex** values defined in this specification and the **InputServerObject** values defined in the Remote Operation (ROP) List and Encoding Protocol Specification, see [MS-OXCROPS] section 3.2.5.1. For details about the relationship between the **OutputHandleIndex** values defined in this specification and the **OutputServerObject** values defined in the ROP List and Encoding Protocol Specification, see [MS-OXCROPS] section 3.2.5.2.

Common **ReturnValues** included in ROP responses are specified in [MS-OXCDATA] section 2.4.

2.2.3.1 FastTransfer Copy Operations

2.2.3.1.1 Download

2.2.3.1.1.1 RopFastTransferSourceCopyTo ROP

The **RopFastTransferSourceCopyTo** ([MS-OXCROPS] section 2.2.12.6) ROP initializes a FastTransfer operation to download content from a given messaging object and its descendant subobjects.

For details about client behaviors related to this ROP, see sections 3.3.5.4.1 and 3.3.5.4.1.1. For details about server behaviors related to this ROP, see sections 3.2.5.4.1 and 3.2.5.4.1.1.

2.2.3.1.1.1.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferSourceCopyTo** ROP ([MS-OXCROPS] section 2.2.12.6).

InputServerObject: MUST be either an **Attachment object**, Message object, or Folder object.

Level (1 byte): An unsigned 8-bit integer. Set to "0" to copy descendant subobjects. Subobjects are only copied when they are not listed in **PropertyTags**. Set to nonzero to exclude all descendant subobjects from being copied. A nonzero value can only be passed if **InputServerObject** is a message or Folder object. The **Level** field MUST be ignored and treated as if it is set to "0" if **InputServerObject** is an Attachment object.

CopyFlags (4 byte): A 32-bit flags structure. This structure defines parameters of the FastTransfer download operation.

The following table defines valid flags for the **CopyFlags** field.

Flag name	Value	Description
Move	0x00000001	MUST NOT be passed if **InputServerObject** is not a folder or a message.<4> If this flag is set, the FastTransfer operation is being configured as a logical part of a larger object move operation, as opposed to a copy operation, and the client will issue further operations such as deleting the moved messages from the source.

Flag name	Value	Description
		If this flag is not set, the FastTransfer operation is not being configured as a logical part of a larger object move operation. For more details, see section 3.2.5.4.1.1.
Unused1	0x00000002	MUST be ignored by the server.
Unused2	0x00000004	MUST be ignored by the server.
Unused3	0x00000008	MUST be ignored by the server.
Unused4	0x00000200	MUST be ignored by the server.
Unused5	0x00000400	MUST be ignored by the server.
BestBody	0x00002000	MUST NOT be passed if **InputServerObject** is not a message. For more details, see section 3.2.5.4.1.1.

SendOptions (1 byte): An 8-bit flag structure. This field defines the parameters of a download operation that relate to data representation.

The following table defines valid flags for the **SendOptions** field.

Flag name	Value	Description
Unicode	0x01	See the following table for all possible combinations of encoding flags (Unicode and ForceUnicode). When used on the **RopSynchronizationConfigure** ROP, MUST match the value of the Unicode **SynchronizationFlag**, as specified in section 2.2.3.2.1.1.2.
UseCpid	0x02	MUST be set for server-to-client-to-server uploads only. For more details about server-to-client-to-server uploads, see section 3.3.5.4.2.1. If this flag is set, the Unicode flag MUST also be set.
ForUpload	0x03	Used in FastTransfer operations only when the client requests a FastTransfer stream with the intent of uploading it immediately to another destination server. The ROP that uses this flag MUST be followed by the **RopTellVersion** ROP. For details about how this affects behaviors of servers and clients, see section 3.3.5.4.2.1.
RecoverMode	0x04	If this flag is set, it indicates that the client supports recovery mode and requests that a server MUST attempt to recover from failures to download changes for individual messages. If this flag is not set, it indicates that the client does not support recovery mode. MUST NOT be set when the **ForUpload** flag is set.
ForceUnicode	0x08	See the following table for all possible combinations of encoding flags (Unicode and ForceUnicode).
PartialItem	0x10	MUST NOT be passed for anything but content synchronization download operations. This flag is set if a client supports partial message downloads.<5>

Flag name	Value	Description
Reserved1	0x20	MUST be set to "0" when sent. The server MUST fail the ROP if this flag is set to "1".
Reserved2	0x40	MUST be set to "0" when sent. The server MUST fail the ROP if this flag is set to "1".

The following table lists all valid combinations of the Unicode | ForceUnicode flags.

Flag name	Description
Neither	String properties MUST be output in the **code page** set on the current connection.
Unicode	String properties MUST be output either in Unicode or in the code page set on the current connection. If the properties are stored in Unicode on the server, the server MUST return the properties in Unicode. If the properties are not stored in Unicode on the server, the server MUST return the properties in the code page set on the current connection.
Unicode | ForceUnicode	String properties MUST be output in Unicode.

PropertyTagCount (2 bytes): An unsigned 16-bit integer. This value specifies the number of structures in the **PropertyTags** field. **PropertyTagCount** is set to zero ("0x0000") if **PropertyTags** is an empty array.

PropertyTags (variable): An array of **PropertyTag** structures. Specifies properties and subobjects, as specified in section 2.2.1.7, to be excluded when copying a messaging object pointed to by the **InputServerObject**. Note that this field MUST NOT be considered when determining what properties and subobjects to copy for descendant subobjects of **InputServerObject**. For more details about the effect of property and subobject filters on download operations, see section 3.2.5.6.

Remarks:

If, for example, the **InputServerObject** is a folder that was opened to show soft deleted messages (such as the **Deleted Items folder**), the scope of an operation that this ROP initiates will only include soft deleted messages. Otherwise, only normal, nondeleted messages will be included. This applies at all levels that are permitted by the **Level** field.

2.2.3.1.1.1.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopFastTransferSourceCopyTo** ROP ([MS-OXCROPS] section 2.2.12.6).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

OutputServerObject: MUST be the FastTransfer download context. MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.1.1.2 RopFastTransferSourceCopyProperties ROP

The **RopFastTransferSourceCopyProperties** ROP ([MS-OXCROPS] section 2.2.12.7) initializes a FastTransfer operation to download content from a specified messaging object and its descendant subobjects.

This ROP is very similar to the **RopFastTransferSourceCopyTo** ROP, with the following exceptions:

- **PropertyTags** specify a list of properties and subobjects to include, as opposed to exclude.
- **BestBody** logic SHOULD NOT be used when copying messages. **BestBody** logic is specified in [MS-OXBBODY]

For details about client behaviors related to this ROP, see sections 3.3.5.4.1 and 3.3.5.4.1.2. For details about server behaviors related to this ROP, see sections 3.2.5.4.1 and 3.2.5.4.1.2.

2.2.3.1.1.2.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferSourceCopyProperties** ROP ([MS-OXCROPS] section 2.2.12.7).

InputServerObject: MUST be either an Attachment object, Message object, or Folder object.

Level (1 byte): An unsigned 8-bit integer. Set to "0" to copy descendant subobjects by using the property list specified in **PropertyTags**. Subobjects are not copied unless listed in **PropertyTags**. Set to nonzero to exclude all descendant subobjects from being copied. A nonzero value can be passed only when **InputServerObject** is a Message object or Folder object. The **Level** field MUST be ignored and treated as if it is set to "0" if **InputServerObject** is an Attachment object.

CopyFlags (1 byte): An 8-bit flag structure. This field defines parameters of the FastTransfer download operation.

The following table defines valid flags for the **CopyFlags** field.

Flag name	Value	Description
Move	0x01	MUST NOT be passed if **InputServerObject** is not a folder or a message. If this flag is set, the FastTransfer operation is being configured as a logical part of a larger object move operation, as opposed to a copy operation, and the client will issue further operations such as deleting the moved messages from the source. If this flag is not set, the FastTransfer operation is not being configured as a logical part of a larger object move operation.
Unused1	0x02	MUST be ignored by the server.
Unused2	0x04	MUST be ignored by the server.
Unused3	0x08	MUST be ignored by the server.

SendOptions (1 byte): An 8-bit flag structure. The possible values for this structure are defined in section 2.2.3.1.1.1.1.

PropertyTagCount (2 bytes): An unsigned 16-bit integer. This value specifies the number of structures in the PropertyTags field. MUST NOT be zero ("0x0000").

PropertyTags (variable): An array of PropertyTag structures. This array specifies the properties and subobjects, as specified in section 2.2.1.7, to copy from the messaging object pointed to by the **InputServerObject**. Note that this field MUST NOT be considered when determining what properties and subobjects to copy for descendant subobjects of **InputServerObject**. For more details about the effect of property and subobject filters on download operations, see section 3.2.5.6.

2.2.3.1.1.2.2 Response Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferSourceCopyProperties** ROP ([MS-OXCROPS] section 2.2.12.7).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

OutputServerObject: MUST be the FastTransfer download context. MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.1.1.3 RopFastTransferSourceCopyMessages ROP

The **RopFastTransferSourceCopyMessages** ROP ([MS-OXCROPS] section 2.2.12.5) initializes a FastTransfer operation on a folder for downloading content and descendant subobjects for messages identified by a given set of IDs.

For more details about client behaviors related to this ROP, see sections 3.3.5.4.1 and 3.3.5.4.1.3. For more details about server behaviors related to this ROP, see sections 3.2.5.4.1 and 3.2.5.4.1.3.

2.2.3.1.1.3.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferSourceCopyMessages** ROP ([MS-OXCROPS] section 2.2.12.5).

InputServerObject: MUST be a Folder object.

MessageIdCount (2 bytes): An unsigned 16-bit integer. This value specifies the number of identifiers in the **MessageIds** field. MUST be greater than zero.

MessageIds (variable): An array of 64-bit identifiers. This list specifies the **MID** values ([MS-OXCDATA] section 2.2.1.2) of the messages to be copied. Messages MUST be contained by a folder identified by **InputServerObject**.

CopyFlags (1 byte): An 8-bit flag structure. This field defines the parameters of the FastTransfer download operation.

The following table defines valid flags for the **CopyFlags** field.

Flag name	Value	Description
Move	0x01	MUST NOT be passed if **InputServerObject** is not a folder. If this flag is set, the FastTransfer operation is being configured as a logical part of a larger object move operation, as opposed to a copy operation, and the client will issue further operations such as deleting the moved messages from the source. If this flag is not set, the FastTransfer operation is not being configured as a logical part of a larger object move operation. For more details, see section 3.2.5.4.1.3.

Flag name	Value	Description
Unused1	0x02	MUST be ignored by the server.
Unused2	0x04	MUST be ignored by the server.
Unused3	0x08	MUST be ignored by the server.
BestBody	0x10	Identifies whether the output **message body (2)** is in the original format or compressed **RTF** format. For more details, see section 3.2.5.4.1.3.
SendEntryId	0x20	If this flag is set, message and change identification information is not removed from output. If this flag is not set, message and change identification information is removed from output.

SendOptions (1 byte): An 8-bit flag structure. The possible values for this structure are defined in section 2.2.3.1.1.1.2.

2.2.3.1.1.3.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopFastTransferSourceCopyMessages** ROP ([MS-OXCROPS] section 2.2.12.5).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

OutputServerObject: MUST be the FastTransfer download context. MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000")

2.2.3.1.1.4 RopFastTransferSourceCopyFolder ROP

The **RopFastTransferSourceCopyFolder** ROP ([MS-OXCROPS] section 2.2.12.4) initializes a FastTransfer operation to download properties and descendant subobjects for a specified folder.

This ROP is very similar to **RopFastTransferSourceCopyTo**, with the following exceptions:

- The type of the **InputServerObject** is limited to a Folder object.
- The FastTransfer stream produced by an operation configured with this ROP wraps folder properties and subobjects with the **topFolder** element (as specified in section 3.1.5.8).
- All properties and contained messages are copied.
- The **CopySubfolders** flag of the **CopyFlag** field indicates whether subfolders are to be copied.
- **BestBody** logic SHOULD NOT be used when copying messages. **BestBody** logic is specified in [MS-OXBBODY].

For details about client behaviors related to this ROP, see sections 3.3.5.4.1 and 3.3.5.4.1.4. For details about server behaviors related to this ROP, see sections 3.2.5.4.1 and 3.2.5.4.1.4.

2.2.3.1.1.4.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferSourceCopyFolder** ROP ([MS-OXCROPS] section 2.2.12.4).

InputServerObject: MUST be a Folder object.

CopyFlags (1 byte): An 8-bit flag structure. This field defines the parameters of the FastTransfer download operation.

The following table defines valid flags for the **CopyFlags** field.

Flag name	Value	Description
Move	0x01	MUST be ignored by the server.<6> If the **Move** flag is set for a download operation, the FastTransfer operation is being configured as a logical part of a larger object move operation, as opposed to a copy operation, and the client will issue further operations such as deleting the moved messages from the source. If the **Move** flag is set for a download operation, the server does not output any objects in a FastTransfer stream that the client does not have permissions to delete. If the **Move** flag is not set, the FastTransfer operation is not being configured as a logical part of a larger object move operation.
Unused1	0x02	MUST be ignored by the server.
Unused2	0x04	MUST be ignored by the server.
Unused3	0x08	MUST be ignored by the server.
CopySubfolders	0x10	Identifies whether the subfolders of the folder specified in the **InputServerObject** are recursively included in the scope. For more details, see section 3.2.5.4.1.4.
NoGhostedContent	0x20	Identifies whether information about the content of a **ghosted** folder is returned. For more details, see section 3.2.5.4.1.4.

SendOptions (1 byte): An 8-bit flag structure. The possible values for this structure are defined in section 2.2.3.1.1.1.2.

2.2.3.1.1.4.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopFastTransferSourceCopyFolder** ROP ([MS-OXCROPS] section 2.2.12.4).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

OutputServerObject: MUST be the FastTransfer download context. MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.1.1.5 RopFastTransferSourceGetBuffer ROP

The **RopFastTransferSourceGetBuffer** ROP ([MS-OXCROPS] section 2.2.12.3) downloads the next portion of a FastTransfer stream that is produced by a previously configured download operation.

RopFastTransferSourceGetBuffer supports packed buffers, as specified in [MS-OXCRPC] section 3.1.7.4.

For more details about client behaviors related to this ROP, see sections 3.3.5.4.1 and 3.3.5.4.1.5. For more details about server behaviors related to this ROP, see sections 3.2.5.4.1 and 3.2.5.4.1.5.

2.2.3.1.1.5.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferSourceGetBuffer** ROP ([MS-OXCROPS] section 2.2.12.3).

InputServerObject: MUST be a download context.

BufferSize (2 bytes): An unsigned 16-bit integer. This field specifies the maximum amount of data (in bytes) to be output in the **TransferBuffer**. For more details, see sections 3.3.5.4.1.5 and 3.2.5.4.1.5.

MaximumBufferSize (2 bytes, optional): An unsigned 16-bit integer that specifies the maximum size limit when the server determines the buffer size.

MUST be present if and only if **BufferSize** is set to a sentinel value of "0xBABE".

2.2.3.1.1.5.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopFastTransferSourceGetBuffer** ROP ([MS-OXCROPS] section 2.2.12.3).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status. For more details, see sections 3.3.5.4.1.5 and 3.2.5.4.1.5.

TransferStatus (2 bytes): A 16-bit enumeration. This field represents the status of the download operation after producing data for the **TransferBuffer** field.<7>

The following table defines valid values for the **TransferStatus** field.

Value	Bit	Description
Error	0x0000	The download stopped because a nonrecoverable error has occurred when producing a FastTransfer stream. The **ReturnValue** field of the ROP response buffer contains a code for that error.
Partial	0x0001	The FastTransfer stream was split, and more data is available. **TransferBuffer** contains incomplete data. For details about where to split FastTransfer streams, see section 2.2.4.1.
Done	0x0003	This was the last portion of the FastTransfer stream.

InProgressCount (2 bytes): An unsigned 16-bit integer. The number of steps that have already been completed in the current operation. Only usable for progress information display.

TotalStepCount (2 bytes): An unsigned 16-bit integer<8> that contains the approximate total number of steps to be completed in the current operation. Only usable for progress information display.

Reserved (1 byte): MUST be set to "0x00" when sending and ignored on receipt.

TransferBufferSize (2 bytes): An unsigned 16-bit integer. This value specifies the size of the **TransferBuffer** field.

TransferBuffer (variable, optional): An array of bytes that contains the next portion of a FastTransfer stream. The syntax of the FastTransfer stream is specified in section 2.2.4. MUST be present if and only if the error code is not **ServerBusy**.

BackoffTime (4 bytes, optional): An unsigned 32-bit integer that contains the time, in milliseconds, that a client waits before retrying the ROP. MUST be present if and only if the error code is **ServerBusy**.

2.2.3.1.1.6 RopTellVersion ROP

The **RopTellVersion** ROP ([MS-OXCROPS] section 2.2.12.8) is used to provide the version of one server to another server that is participating in the server-to-client-to-server upload, as specified in section 3.3.5.4.2.1.

For more details about client behaviors related to this ROP, see sections 3.3.5.4.1 and 3.3.5.4.1.6. For more details about server behaviors related to this ROP, see sections 3.2.5.4.1 and 3.2.5.4.1.6.

2.2.3.1.1.6.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopTellVersion** ROP ([MS-OXCROPS] section 2.2.12.8).

Version (6 bytes): An array of three unsigned 16-bit integers. This array contains the version information for another server that is participating in the server-to-client-to-server upload. The format of this structure is the same as that specified in [MS-OXCRPC] section 3.1.9.

2.2.3.1.1.6.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopTellVersion** ROP ([MS-OXCROPS] section 2.2.12.8).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

2.2.3.1.2 Upload

2.2.3.1.2.1 RopFastTransferDestinationConfigure ROP

The **RopFastTransferDestinationConfigure** ([MS-OXCROPS] section 2.2.12.1) ROP initializes a FastTransfer operation for uploading content encoded in a client-provided FastTransfer stream into a mailbox.

For more details about client behaviors related to this ROP, see sections 3.3.5.4.2 and 3.3.5.4.2.2. For more details about server behaviors related to this ROP, see section 3.2.5.4.2.1.

2.2.3.1.2.1.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferDestinationConfigure** ROP ([MS-OXCROPS] section 2.2.12.1).

InputServerObject: MUST be either an Attachment object, Message object, or Folder object.

SourceOperation (1 byte): An 8-bit enumeration. This enumeration is used to specify the type of data in a FastTransfer stream that is uploaded by using the

RopFastTransferDestinationPutBuffer ROP on the FastTransfer upload context that is returned in the **OutputServerObject** field.

The following table defines the **SourceOperation** enumeration values and the associated root elements in the FastTransfer stream.

SourceOperation enumeration value	Root element in FastTransfer stream	Conditions
CopyTo **CopyProperties**	**folderContent**	**InputServerObject** is a Folder object.
	messageContent	**InputServerObject** is a Message object.
	attachmentContent	**InputServerObject** is an Attachment object.
CopyMessages	**messageList**	Always.
CopyFolder	**topFolder**	Always.

The following table defines the **SourceOperation** ordinal values.

SourceOperation enumeration value	Ordinal value	Corresponding ROP of the FastTransfer download*
CopyTo	0x01	**RopFastTransferSourceCopyTo**
CopyProperties	0x02	**RopFastTransferSourceCopyProperties**
CopyMessages	0x03	**RopFastTransferSourceCopyMessages**
CopyFolder	0x04	**RopFastTransferSourceCopyFolder**

*If the FastTransfer stream to be uploaded was produced by a FastTransfer download operation, the **SourceOperation** value MUST correspond to the **RopFastTransferSourceCopy*** ROP that was used to configure the download operation.

CopyFlags (1 byte): 8-bit flag structure. This field defines the parameters of the FastTransfer upload operation.

The following table defines valid flags for the **CopyFlags** field.

Flag name	Value	Description
Move	0x01	MUST NOT be passed if **InputServerObject** is not a folder or a message. If this flag is set, the FastTransfer operation is being configured as a logical part of a larger object move operation, as opposed to a copy operation, and the client will issue further operations such as deleting the moved messages from the source. If this flag is not set, the FastTransfer operation is not being configured as a logical part of a larger object move operation.

2.2.3.1.2.1.2 Response Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferDestinationConfigure** ROP ([MS-OXCROPS] section 2.2.12.1).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

OutputServerObject: MUST be the FastTransfer upload context. MUST be present if and only if **ReturnValue** equals **Success** (0x00000000).

2.2.3.1.2.2 RopFastTransferDestinationPutBuffer ROP

The **RopFastTransferDestinationPutBuffer** ROP ([MS-OXCROPS] section 2.2.12.2) uploads the next portion of an input FastTransfer stream for a previously configured FastTransfer upload operation.

For more details about server behaviors related to this ROP, see section 3.2.5.4.2.2.

2.2.3.1.2.2.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopFastTransferDestinationPutBuffer** ROP ([MS-OXCROPS] section 2.2.12.2).

InputServerObject: MUST be the FastTransfer upload context.

TransferDataSize (2 bytes): An unsigned 16-bit integer. This value specifies the size of the **TransferData** field. MUST NOT be zero ("0x0000"). The maximum data size is limited to the available space allowed by the underlying transport used by the **EcDoRpcExt2** RPC ([MS-OXCRPC] section 3.1.4.12).

TransferData (variable): An array of bytes. This array contains the data to be uploaded to the destination **FastTransfer** object and contains the next portion of a FastTransfer stream. The syntax of the FastTransfer stream is specified in section 2.2.4.

2.2.3.1.2.2.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopFastTransferDestinationPutBuffer** ROP ([MS-OXCROPS] section 2.2.12.2).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

TransferStatus (2 bytes): A 16-bit enumeration. Clients MUST ignore the value of this field.

InProgressCount (2 bytes): An unsigned 16-bit integer that specifies the number of steps that have been completed in the current operation. The server SHOULD set this field to "0x0000". This field is usable only for progress information display.

TotalStepCount (2 bytes): An unsigned 16-bit integer that contains the approximate total number of steps to be completed in the current operation. The server SHOULD set this field to "0x0000". This field is usable only for progress information display.

Reserved (1 byte): MUST be set to "0x00" when sending, and MUST be ignored on receipt.

BufferUsedSize (2 bytes): An unsigned 16-bit integer. This value is the buffer size that was used. Can be less than **TransferDataSize** if and only if a ROP failed and **ReturnValue** is not equal to **Success** (0x00000000).

2.2.3.2 Incremental Change Synchronization

2.2.3.2.1 Download

2.2.3.2.1.1 RopSynchronizationConfigure ROP

The **RopSynchronizationConfigure** ROP ([MS-OXCROPS] section 2.2.13.1) is used to define the scope and parameters of the synchronization download operation.

The synchronization scope determines the boundaries of a synchronization operation, and is defined by the following:

- The type of objects considered for synchronization (folders for hierarchy synchronization operations, and messages for content synchronization operations).
- A folder that contains these objects as children (contents) or descendants (hierarchy).
- A restriction on messages within that folder (contents).

For more details, see section 3.3.5.2.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.1 and 3.3.5.5.1.1. For more details about server behaviors related to this ROP, see section 3.2.5.5.1.1.

2.2.3.2.1.1.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationConfigure** ROP ([MS-OXCROPS] section 2.2.13.1).

InputServerObject: MUST be a Folder object that contributes to the synchronization scope.

SynchronizationType (1 byte): An 8-bit enumeration that defines the type of synchronization requested: contents or hierarchy. This field contributes to the synchronization scope.

The following table defines valid flags for the **SynchronizationType** field.

Flag name	Value	Description
Contents	0x01	Indicates a content synchronization operation.
Hierarchy	0x02	Indicates a hierarchy synchronization operation.

SendOptions (1 byte): An 8-bit enumeration that identifies options for sending the data. For details about the possible values for this enumeration, see section 2.2.3.1.1.1.2.

SynchronizationFlag (2 bytes): A 16-bit flag structure that defines the parameters of the synchronization operation.

The following table defines valid flags for the **SynchronizationFlag** field.

Flag name	Value	Description
Unicode	0x0001	Indicates whether the client supports Unicode. For more details, see section 3.2.5.5.1.1. This flag MUST match the value of the Unicode flag from the **SendOptions** field.
NoDeletions	0x0002	Indicates how the server downloads information about item deletions. For more details, see section 3.2.5.5.1.1. The client MAY implement this flag.<9>
IgnoreNoLongerInScope	0x0004	Indicates whether the server downloads information about messages that went out of scope. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation. The client MAY implement this flag.<10>
ReadState	0x0008	Indicates whether the server downloads information about changes to the read state of messages. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.
FAI	0x0010	Indicates whether the server downloads information about changes to FAI messages. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.
Normal	0x0020	Indicates whether the server downloads information about changes to normal messages. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.
OnlySpecifiedProperties	0x0080	Indicates whether the server limits or excludes properties and subobjects output to the properties listed in **PropertyTags**. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.
NoForeignIdentifiers	0x0100	Identifies whether the server ignores any persisted values for the **PidTagSourceKey** property (section 2.2.1.2.5) and **PidTagParentSourceKey** property (section 2.2.1.2.6) when producing output for folder and message changes. For more details, see section 3.2.5.5.1.1. Clients SHOULD set this flag. For more details about possible issues if this flag is not set, see section 3.3.5.1.3.
Reserved	0x1000	MUST be set to "0" when sending. The client MAY implement this flag.<11>
BestBody	0x2000	Identifies how the server outputs message bodies (2). For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.

Flag name	Value	Description
IgnoreSpecifiedOnFAI	0x4000	Indicates whether the server outputs properties and subobjects of FAI messages. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.
Progress	0x8000	Indicates whether the server injects progress information into the output FastTransfer stream. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but content synchronization download operation. This flag is in addition to the means of progress reporting available through the **RopFastTransferSourceGetBuffer** ROP results.

RestrictionDataSize (2 bytes): An unsigned 16-bit integer that specifies the length of the **RestrictionData** field. MUST be set to "0x0000" if **SynchronizationType** is set to **Hierarchy** ("0x02").

RestrictionData (variable): The variable-length restriction structure, which is used to select the data to be synchronized. This value contributes to the synchronization scope. This field is used in content synchronization operations only. The value MUST be set to "0" if **SynchronizationType** is set to Hierarchy ("0x02"). For more details about restrictions, see [MS-OXCDATA].

SynchronizationExtraFlag (4 bytes): A 32-bit flag structure.

The following table defines valid flags for the **SynchronizationExtraFlag** field.

Flag name	Value	Description
Eid	0x00000001	Indicates whether the server includes the **PidTagFolderId** (section 2.2.1.2.2) or **PidTagMid** (section 2.2.1.2.1) properties in the folder change or message change header. For more details, see section 3.2.5.5.1.1.
MessageSize	0x00000002	Indicates whether the server includes the **PidTagMessageSize** property (section 2.2.1.5) in the message change header. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.
CN	0x00000004	Indicates whether the server includes the **PidTagChangeNumber** property (section 2.2.1.2.3) in the message change header. For more details, see section 3.2.5.5.1.1.
OrderByDeliveryTime	0x00000008	Indicates whether the server sorts messages by their delivery time. For more details, see section 3.2.5.5.1.1. MUST NOT be passed for anything but a content synchronization download operation.

PropertyTagCount (2 bytes): An unsigned 16-bit integer that specifies the number of **PropertyTag** structures in **PropertyTags**. **PropertyTagCount** is set to zero ("0x0000") if **PropertyTags** is an empty array.

PropertyTags (variable): An array of **PropertyTag** structures that specifies properties and subobjects, as specified in section 2.2.1.7, to exclude or include.

In addition to regular property tags, this field can contain property tags for the properties that denote message subobjects, as specified in section 2.2.1.7. For more details, see section 3.2.5.5.1.1.

2.2.3.2.1.1.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationConfigure** ROP ([MS-OXCROPS] section 2.2.13.1).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the status of the ROP execution.

OutputServerObject: This value MUST be the synchronization download context. This value MUST be present if and only if **ReturnValue** is **Success** ("0x00000000").

2.2.3.2.2 Uploading State

2.2.3.2.2.1 RopSynchronizationUploadStateStreamBegin ROP

The **RopSynchronizationUploadStateStreamBegin** ROP ([MS-OXCROPS] section 2.2.13.9) initiates the upload of an ICS state property into the synchronization context.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.2 and 3.3.5.5.2.1. For more details about server behaviors related to this ROP, see section 3.2.5.5.2.1.

2.2.3.2.2.1.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationUploadStateStreamBegin** ROP ([MS-OXCROPS] section 2.2.13.9).

InputServerObject: MUST be a synchronization context.

StateProperty (4 bytes): A 32-bit **PropertyTag** structure. Valid input is restricted to the property tags of the ICS state properties: **PidTagIdsetGiven** (section 2.2.1.1.1), **PidTagCnsetSeen** (section 2.2.1.1.2), **PidTagCnsetSeenFAI** (section 2.2.1.1.3), and **PidTagCnsetRead** (section 2.2.1.1.4).

TransferBufferSize (4 bytes): An unsigned 32-bit integer. This value specifies the size of the stream to be uploaded by the **RopSynchronizationUploadStateStreamContinue** ROP.

2.2.3.2.2.1.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationUploadStateStreamBegin** ROP ([MS-OXCROPS] section 2.2.13.9).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

2.2.3.2.2.2 RopSynchronizationUploadStateStreamContinue ROP

The **RopSynchronizationUploadStateStreamContinue** ROP ([MS-OXCROPS] section 2.2.13.10) continues to upload an ICS state property value into the synchronization context.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.2 and 3.3.5.5.2.2. For more details about server behaviors related to this ROP, see section 3.2.5.5.2.2.

2.2.3.2.2.2.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationUploadStateStreamContinue** ROP ([MS-OXCROPS] section 2.2.13.10).

InputServerObject: MUST be a synchronization context.

StreamDataSize (4 bytes): An unsigned 32-bit integer. This value specifies the size of the **StreamData** field. MUST NOT be set to zero ("0x00000000").

StreamData (variable): This array contains the state stream data to be uploaded.

2.2.3.2.2.2.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationUploadStateStreamContinue** ROP ([MS-OXCROPS] section 2.2.13.10).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

2.2.3.2.2.3 RopSynchronizationUploadStateStreamEnd ROP

The **RopSynchronizationUploadStateStreamEnd** ROP ([MS-OXCROPS] section 2.2.13.11) concludes the upload of an ICS state property value into the synchronization context.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.2 and 3.3.5.5.2.3. For more details about server behaviors related to this ROP, see section 3.2.5.5.2.3.

2.2.3.2.2.3.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationUploadStateStreamEnd** ROP ([MS-OXCROPS] section 2.2.13.11).

InputServerObject: MUST be a synchronization context.

2.2.3.2.2.3.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationUploadStateStreamEnd** ROP ([MS-OXCROPS] section 2.2.13.11).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

2.2.3.2.3 Downloading State

2.2.3.2.3.1 RopSynchronizationGetTransferState ROP

The **RopSynchronizationGetTransferState** ([MS-OXCROPS] section 2.2.13.8) ROP creates a FastTransfer download context for a snapshot of the **checkpoint ICS state** of the operation identified by the given synchronization download context or synchronization upload context. This ROP will return the initial ICS state for the download context until the end of the FastTransfer stream has been downloaded, but it will return the checkpoint ICS state that is reflective of the

current snapshot for an upload context. After the download is complete, this ROP will return the final ICS state.<12>

For more details about client behaviors related to this ROP, see sections 3.3.5.5.3 and 3.3.5.5.3.1. For more details about server behaviors related to this ROP, see section 3.2.5.5.3.1.

2.2.3.2.3.1.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationGetTransferState** ([MS-OXCROPS] section 2.2.13.8).

InputServerObject: MUST be either a synchronization download context or synchronization upload context.

2.2.3.2.3.1.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationGetTransferState** ([MS-OXCROPS] section 2.2.13.8).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

OutputServerObject: MUST be the FastTransfer download context for the ICS state. MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.2.4 Upload

2.2.3.2.4.1 RopSynchronizationOpenCollector ROP

The **RopSynchronizationOpenCollector** ROP ([MS-OXCROPS] section 2.2.13.7) configures the synchronization upload operation and returns a handle to a synchronization upload context.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.3. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.1.

2.2.3.2.4.1.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationOpenCollector** ROP ([MS-OXCROPS] section 2.2.13.7).

InputServerObject: MUST be a Folder object that contributed to the synchronization scope that corresponds to the initial ICS state to be uploaded, as specified in section 3.3.5.2.

IsContentsCollector (1 byte): An 8-bit **PtypBoolean** ([MS-OXCDATA] section 2.11.1) value. This value is "0x01" (nonzero) if a synchronization upload is requested for contents of folders, or "0x00" if a synchronization upload is requested for the hierarchy of the folder contents.

2.2.3.2.4.1.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationOpenCollector** ROP ([MS-OXCROPS] section 2.2.13.7).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

OutputServerObject: MUST be the synchronization upload context. MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.2.4.2 RopSynchronizationImportMessageChange

The **RopSynchronizationImportMessageChange** ROP ([MS-OXCROPS] section 2.2.13.2) is used to import new messages or changes to existing messages into the server replica. When there are changes to existing messages, the entire changed message MUST be uploaded.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.4. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.2.

2.2.3.2.4.2.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationImportMessageChange** ROP ([MS-OXCROPS] section 2.2.13.2).

InputServerObject: MUST be the synchronization upload context configured for the collection of changes to content.

ImportFlag (1 byte): An 8-bit flag structure.

The following table defines valid flags for the **ImportFlag** field.

Flag name	Value	Description
Associated	0x10	If this flag is set, the message being imported is an FAI message. If this flag is not set, the message being imported is a normal message.
FailOnConflict<13>	0x40	Specifies whether the server accepts conflicting versions of a particular message. For more details, see section 3.2.5.5.4.2.

PropertyValueCount (2 bytes): An unsigned 16-bit integer. This value specifies the number of structures in the **PropertyValues** field. MUST NOT be zero ("0x0000").

PropertyValues (variable): An array of **TaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.4). These values are used to specify extra properties on the message; these are properties that cannot be set by using the **RopSetProperties** ROP.

The following table lists the restrictions that exist for properties passed in the **PropertyValues** field.

Property	Restrictions	Comments
PidTagSourceKey (section 2.2.1.2.5)	Required Fixed position	The **GID** value ([MS-OXCDATA] section 2.2.1.3) of the message being uploaded in the local replica.
PidTagLastModificationTime ([MS-OXPROPS] section 2.861)	Required Fixed position	None.
PidTagChangeKey (section 2.2.1.2.7)	Required Fixed position	The **XID**, as specified in section 2.2.2.2, of a change of a message being uploaded in a local replica. For details about how clients can generate this value, see section 3.1.5.3.

Property	Restrictions	Comments
PidTagPredecessorChangeList (section 2.2.1.2.8)	Required Fixed position	None.
< other properties >	*Prohibited*	None.

2.2.3.2.4.2.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationImportMessageChange** ROP ([MS-OXCROPS] section 2.2.13.2).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status. For details about the common return values for **RopSynchronizationImport*** ROPs that require special processing, see section 3.3.5.5.4. The following table contains one additional return value.

Return value name	Value	Description
SyncConflict	0x80040802	A conflict has occurred and conflict resolution failed. No data was imported.

OutputServerObject: MUST be the Message object into which the client will upload the rest of the message changes. MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

MessageId (8 bytes): A 64-bit identifier that specifies the **MID** value ([MS-OXCDATA] section 2.2.1.2) of the message that was imported. MUST be set to "0x0000000000000000". MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.2.4.3 RopSynchronizationImportHierarchyChange ROP

The **RopSynchronizationImportHierarchyChange** ROP ([MS-OXCROPS] section 2.2.13.4) is used to import new folders, or changes to existing folders, into the server replica.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.5. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.3.

2.2.3.2.4.3.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationImportHierarchyChange** ROP ([MS-OXCROPS] section 2.2.13.4).

InputServerObject: MUST be the synchronization upload context configured to collect changes to the hierarchy.

HierarchyValueCount (2 bytes): An unsigned 16-bit integer. This value specifies the number of structures in the **HierarchyValues** field. MUST NOT be zero ("0x0000").

HierarchyValues (variable): An array of **TaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.4). These values are used to specify folder hierarchy properties, which determine the location of the folder within the hierarchy. The following table lists the restrictions that exist on the **HierarchyValues** field.

Property	Restrictions	Comments
PidTagParentSourceKey (section 2.2.1.2.6)	Required Fixed position	Can be zero-length to identify a folder for which a synchronization upload context was opened.
PidTagSourceKey (section 2.2.1.2.5)	Required Fixed position	The **GID** value ([MS-OXCDATA] section 2.2.1.3) of the folder being uploaded in the local replica.
PidTagLastModificationTime ([MS-OXPROPS] section 2.861)	Required Fixed position	None.
PidTagChangeKey (section 2.2.1.2.7)	Required Fixed position	The **XID**, as specified in section 2.2.2.2, of a change being uploaded in a local replica. For details about how clients can generate the **PidTagChangeKey** value, see section 3.1.5.3.
PidTagPredecessorChangeList (section 2.2.1.2.8)	Required Fixed position	None.
PidTagDisplayName ([MS-OXCFOLD] section 2.3.2.2.3)	Required Fixed position	Value MUST be a nonempty string.
< other properties >	*Prohibited*	None.

PropertyValueCount (2 bytes): An unsigned 16-bit integer. This value specifies the number of structures in the PropertyValues field. MUST NOT be zero ("0x0000").

PropertyValues (variable): An array of **TaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.1). These values are used to specify folder properties.

2.2.3.2.4.3.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationImportHierarchyChange** ROP ([MS-OXCROPS] section 2.2.13.4).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status. For details about common return values of the **RopSynchronizationImport*** ROPs that require special processing, see section 3.3.5.5.4.

FolderId (8 bytes): A 64-bit identifier. The **FID** value ([MS-OXCDATA] section 2.2.1.1) of the folder that was imported. MUST be set to "0x0000000000000000". MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.2.4.4 RopSynchronizationImportMessageMove ROP

The **RopSynchronizationImportMessageMove** ROP ([MS-OXCROPS] section 2.2.13.6) imports information about moving a message between two existing folders within the same mailbox.

To move folders within a mailbox, use the **RopSynchronizationImportHierarchyChange** ROP.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.6. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.4.

2.2.3.2.4.4.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationImportMessageMove** ROP ([MS-OXCROPS] section 2.2.13.6).

InputServerObject: MUST be the synchronization upload context configured for collecting changes to the contents of the message move destination folder.

SourceFolderIdSize (4 bytes): An unsigned 32-bit integer. This value specifies the size of the **SourceFolderId** field. MUST NOT be zero ("0x00000000").

SourceFolderId (variable): An array of bytes. This value contains a serialized representation of the **GID** value ([MS-OXCDATA] section 2.2.1.3) that represents the **PidTagSourceKey** property (section 2.2.1.2.5) value of the source folder. The source folder MUST be in the same mailbox as the destination folder specified in **InputServerObject**.

SourceMessageIdSize (4 bytes): An unsigned 32-bit integer. This value specifies the size of the **SourceMessageId** field. MUST NOT be zero ("0x00000000").

SourceMessageId (variable): An array of bytes. This value contains a serialized representation of the **GID** value that represents the **PidTagSourceKey** property of the message in the source folder, identified by **SourceFolderId** field.

PredecessorChangeListSize (4 bytes): An unsigned 32-bit integer. This value specifies the size of the **PredecessorChangeList** field. MUST NOT be zero ("0x00000000").

PredecessorChangeList (variable): An array of bytes. This value contains a serialized representation of the **PidTagPredecessorChangeList** property (section 2.2.1.2.8) value in the local replica of the message being moved.

DestinationMessageIdSize (4 bytes): An unsigned 32-bit integer. This value specifies the size of the **DestinationMessageId** field. MUST NOT be zero ("0x00000000").

DestinationMessageId (variable): An array of bytes. This value contains a serialized representation of the **GID** value that represents the **PidTagSourceKey** property of the message in the destination folder. For details about why the **DestinationMessageId** value is different from the **SourceMessageId** value, see section 3.1.5.3.

ChangeNumberSize (4 bytes): An unsigned 32-bit integer. This value specifies the size of the **ChangeNumber** field. MUST NOT be zero ("0x00000000").

ChangeNumber (variable): An array of bytes. This value contains a serialized representation of the **XID**, as specified in section 2.2.2.2, that represents the **PidTagChangeKey** property (section 2.2.1.2.7) of the message in the destination folder.

2.2.3.2.4.4.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationImportMessageMove** ROP ([MS-OXCROPS] section 2.2.13.6).

Return value (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status. For details about the common return values of the **RopSynchronizationImport*** ROPs that require special processing, see section 3.3.5.5.4. The following table contains additional return values.

Return value name	Value	Description
NewerClientChange	0x00040821	The ROP succeeded, but the server replica had an older version of a message than the local replica. **ChangeNumber** and **PredecessorChangeList** were not applied to the destination message.

The complete list of error codes is specified in [MS-OXCDATA] section 2.4.

MessageId (8 bytes): A 64-bit identifier. The **MID** value ([MS-OXCDATA] section 2.2.1.2) of the moved message in a destination folder. MUST be set to "0x0000000000000000". MUST be present if and only if **ReturnValue** equals **Success** ("0x00000000").

2.2.3.2.4.5 RopSynchronizationImportDeletes ROP

The **RopSynchronizationImportDeletes** ROP ([MS-OXCROPS] section 2.2.13.5) imports deletions of messages or folders into the server replica.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.7. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.5.

2.2.3.2.4.5.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationImportDeletes** ROP ([MS-OXCROPS] section 2.2.13.5).

InputServerObject: MUST be the synchronization upload context. The type of synchronization upload context MUST correspond to the **ImportDeleteFlags** field.

ImportDeleteFlags (1 byte): An 8-bit flag structure.

The following table defines valid flags for the **ImportDeleteFlags** field.

Flag name	Value	Description
Hierarchy	0x01	If this flag is set, folder deletions are being imported. If this flag is not set, message deletions are being imported.
HardDelete<14>	0x02	If this flag is set, hard deletions are being imported. If this flag is not set, hard deletions are not being imported.

PropertyValueCount (2 bytes): An unsigned 16-bit integer. This value specifies the number of structures present in the **PropertyValues** field.

PropertyValues (variable): An array of **TaggedPropertyValue** structures ([MS-OXCDATA] section 2.11.4). MUST NOT be NULL. The value of this field is used to specify the folders or messages to be deleted.

The following table defines the restrictions that exist on the **PropertyValues** field.

Property	Restrictions	Comments
[MVBinary] 0x00001102	Required Fixed position	An array of serialized **GID** values ([MS-OXCDATA] section 2.2.1.3) that represent the objects to be deleted.

Property	Restrictions	Comments
< other properties >	*Prohibited*	None.

2.2.3.2.4.5.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSynchronizationImportDeletes** ROP ([MS-OXCROPS] section 2.2.13.5).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status. For details about common return values for **RopSynchronizationImport*** ROPs that require special processing, see section 3.3.5.5.4.

2.2.3.2.4.6 RopSynchronizationImportReadStateChanges ROP

The **RopSynchronizationImportReadStateChanges** ROP ([MS-OXCROPS] section 2.2.13.3) imports message read state changes into the server replica.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.8. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.6.

2.2.3.2.4.6.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationImportReadStateChanges** ROP ([MS-OXCROPS] section 2.2.13.3).

InputServerObject: MUST be the synchronization upload context configured to collect changes to content.

MessageReadStateSize (2 bytes): An unsigned 16-bit integer. This value specifies the size in bytes of the **MessageReadStates** field. MUST NOT be zero ("0x0000").

MessageReadStates (variable): An array of **MessageReadState** structures ([MS-OXCROPS] section 2.2.13.3.1.1) — one per each message that is changing its read state — that consist of the following:

- **MessageIdSize (2 bytes):** An unsigned 16-bit integer. This value specifies the size of the **MessageId** field. MUST NOT be zero ("0x0000").
- **MessageId (variable):** An array of bytes. Contains the **XID**, as specified in section 2.2.2.2, that represents the **PidTagSourceKey** property (section 2.2.1.2.5) for a message that is changing its read state.
- **MarkAsRead (1 byte):** An 8-bit **PtypBoolean** ([MS-OXCDATA] section 2.11.1). This value specifies whether to mark the message as read ("0x01") or unread ("0x00").

MID values ([MS-OXCDATA] section 2.2.1.2) of FAI messages in **MessageReadStates** are ignored.

2.2.3.2.4.6.2 Response Buffer

The following descriptions define valid fields for the request buffer of the **RopSynchronizationImportReadStateChanges** ROP ([MS-OXCROPS] section 2.2.13.3).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status. For details about common return values for the **RopSynchronizationImport*** ROPs that require special processing, see section 3.3.5.5.4.

2.2.3.2.4.7 RopGetLocalReplicaIds ROP

The **RopGetLocalReplicaIds** ROP ([MS-OXCROPS] section 2.2.13.13) allocates a range of internal identifiers (2) for the purpose of assigning them to client-originated objects in a local replica. For more details about client-assigned internal identifiers (2), see section 3.3.5.1.1.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.9. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.7.

2.2.3.2.4.7.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopGetLocalReplicaIds** ROP ([MS-OXCROPS] section 2.2.13.13).

InputServerObject: MUST be a **Logon** object.

IdCount (4 bytes): An unsigned 32-bit integer. This value specifies the number of IDs to be allocated.

2.2.3.2.4.7.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopGetLocalReplicaIds** ROP ([MS-OXCROPS] section 2.2.13.13).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

REPLGUID (16 bytes): A GUID that specifies the REPLGUID shared by all allocated IDs. MUST be present if and only if **ReturnValue** equals Success ("0x00000000").

GlobalCount (6 bytes): An array of bytes. This array specifies the value of the GLOBCNT field for the first allocated ID in the allocated set of [**GlobalCount**, **GlobalCount** + **IdCount** – 1]. MUST be present if and only if **ReturnValue** equals Success ("0x00000000").

2.2.3.2.4.8 RopSetLocalReplicaMidsetDeleted ROP

The **RopSetLocalReplicaMidsetDeleted** ROP ([MS-OXCROPS] section 2.2.13.12) identifies that a set of IDs either belongs to deleted messages in the specified folder or will never be used for any messages in the specified folder.

All of the IDs contained in **LongTermIdRanges** structures MUST have been obtained previously by using the **RopGetLocalReplicaIds** ROP.

RopSetLocalReplicaMidsetDeleted does not deallocate IDs; it only reports that they cannot be used within a given folder. For details about using the **RopSetLocalReplicaMidsetDeleted** ROP, see section 3.2.5.5.4.8.

For more details about client behaviors related to this ROP, see sections 3.3.5.5.4 and 3.3.5.5.4.10. For more details about server behaviors related to this ROP, see section 3.2.5.5.4.8.

2.2.3.2.4.8.1 Request Buffer

The following descriptions define valid fields for the request buffer of the **RopSetLocalReplicaMidsetDeleted** ROP ([MS-OXCROPS] section 2.2.13.12).

InputServerObject: MUST be a Folder object.

DataSize (2 bytes): An unsigned 16-bit integer. This value specifies the size of both the **LongTermIdRangeCount** and **LongTermIdRanges** fields. MUST NOT be zero ("0x0000").

LongTermIdRangeCount (4 bytes): An unsigned 32-bit integer. This value specifies the number of structures in the **LongTermIdRanges** field. MUST NOT be zero ("0x00000000").

LongTermIdRanges (variable): An array of **LongTermIdRange** structures. Each **LongTermIdRange** structure defines a range of IDs, which are reported as unused or deleted. Consists of the following:

- **MinLongTermId (24 bytes):** A LongTermID structure that defines the ID by using the minimum value of a GLOBCNT part that belongs to a range.
- **MaxLongTermId (24 bytes):** A LongTermID structure that defines the ID by using the maximum value of a GLOBCNT part that belongs to a range.

The REPLGUID parts of **MinLongTermId** and **MaxLongTermId** MUST be the same.

2.2.3.2.4.8.2 Response Buffer

The following descriptions define valid fields for the response buffer of the **RopSetLocalReplicaMidsetDeleted** ROP ([MS-OXCROPS] section 2.2.13.12).

ReturnValue (4 bytes): An unsigned 32-bit integer. This value represents the ROP execution status.

2.2.4 FastTransfer Stream

The information set encoded in a FastTransfer stream depends on the type and parameters of the operation that produces it, as specified in section 3.1.5.8. Parsing (syntactic analysis) of the stream can be done without knowing what operation produced it.

At a high level, the FastTransfer stream contains serialized mailbox data and **markers**. Note that markers are not properties and can never have a value, although they are specified in [MS-OXPROPS] and have the same syntax as property tags. The complete list of markers is specified in section 2.2.4.1.4.

Section 2.2.4.1 and section 2.2.4.2 contain an **Augmented Backus-Naur Form (ABNF)** like description of the tokenized FastTransfer stream structure. The description uses the conventions specified in [RFC5234], except for the following:

- Names enclosed in curly brackets indicate terminal tokens that are serializations of simple types, as specified in section 2.2.4.1.3. They can be followed by *prose* definitions that add restrictions to disambiguate the lexical analysis.
- For display purposes, indented lines represent a continuation of the lines that precede them.

Despite their name, FastTransfer streams are not represented as Stream objects, and they can only be manipulated by using the **RopFastTransferSourceGetBuffer** ROP for download operations and

RopFastTransferDestinationPutBuffer for upload operations. For more details about how FastTransfer streams are produced and processed by ROPs, see section 3.1.5.8.

2.2.4.1 Lexical structure

Lexical structure of the FastTransfer stream is essential to let its producers and consumers agree on rules that govern splitting of the stream into sequential buffers retrieved by using the **RopFastTransferSourceGetBuffer** ROP or supplied through the **RopFastTransferDestinationPutBuffer** ROP. It is also beneficial for an explanation of the protocol, as it separates matters of data serialization and deserialization (lexical analysis) from data and data organization (syntactical analysis), and from its mapping to mailbox concepts (semantics).

The lexical structure of a FastTransfer stream is as follows:

```
stream            = 1*element
element           = marker / propValue
marker            = PtypInteger32 <from the table in 2.2.4.1.4>
propValue         = fixedPropType propInfo fixedSizeValue
propValue         =/ varPropType propInfo length varSizeValue
propValue         =/ mvPropType
                        propInfo
                        length
                        *( fixedSizeValue / length varSizeValue )
propInfo          = taggedPropId / ( namedPropId namedPropInfo )
fixedSizeValue    = PtypInteger16 / PtypInteger32 / PtypFloating32
                    / PtypFloating64 / PtypCurrency / PtypFloatingTime
                    / PtypBoolean / PtypInteger64 / PtypTime
                    / PtypGuid
varSizeValue      = PtypString / PtypString8 / PtypServerId
                    / PtypBinary / PtypObject


namedPropInfo     = propertySet
                     ((%x00 dispid)
                     / (%x01 name))
propertySet        = PtypGuid
dispid             = PtypInteger32
name               = PtypString
groupTypedPropInfo       = proptype
                            ( taggedPropId / ( namedPropId groupNamedPropInfo ) )
groupNamedPropInfo  = PropertySet
                     ((%x00000000 dispid)
                     / (%x01000000 name))
namedPropId       = propertyId
                     <Greater or equal to 0x8000>
propertyId        = PtypInteger16
taggedPropId       = propertyId
                     <less than 0x8000>
length            = PtypInteger32 <MUST be greater than 0>
propType          = fixedPropType / varPropType / mvPropType
fixedPropType     = PtypInteger16
varPropType       = PtypInteger16
mvPropType        = PtypInteger16
```

For more details about the **fixedPropType**, **varPropType**, and **mvPropType** **property types**, see section 2.2.4.1.1.

The lexical structure of the FastTransfer adheres to the following guidelines:

- **Camel-cased** names are nonterminal syntactic elements, as specified in [RFC5234] section 2.3.
- **Pascal-cased** names with a **Ptyp** prefix are any value of that type serialized as specified in section 2.2.4.1.3.

A FastTransfer stream can be larger than a single buffer. The server MUST split the stream when it cannot fit into a single buffer. If a split is required, the stream MUST be split either between two atoms or at any point inside a **varSizeValue**. A stream MUST NOT be split within a single atom. The lexical structure of an atom is as follows:

```
atom                = marker
                    / propDef
                    / fixedSizeValue
                    / length
propDef             = ( propType propInfo )
```

2.2.4.1.1 fixedPropType, varPropType, mvPropType

Property types supported in FastTransfer streams are a subset of those defined in [MS-OXCDATA] section 2.11.1.

Property type	Description
fixedPropType	Property type value of any type that has a fixed length, as specified in [MS-OXCDATA] section 2.11.1.
varPropType	Property type value of either **PtypString**, **PtypString8** or **PtypBinary**, **PtypServerId**, or **PtypObject** ([MS-OXCDATA] section 2.11.1).
mvPropType	Property type value of any multi-valued property type (starts with **PtypMultiple** ([MS-OXCDATA] section 2.11.1)), whose base type is either a valid **fixedPropType** or a valid **varPropType**.

2.2.4.1.2 propValue

The **propValue** element represents the identification and a value of a property or a **meta-property**.

The **fixedSizeValue** or **varSizeValue** lexemes contained in a **propValue** represent a value of the property and MUST be serializations of a **base property type** for a property type specified with contained **fixedPropType**, **varPropType**, or **mvPropType** values.

2.2.4.1.3 Serialization of Simple Types

Serialization of simple types in FastTransfer streams is identical to serialization of property values as specified [MS-OXCDATA], with the following exceptions:

Property type name	Difference in serialization
PtypBoolean ([MS-OXCDATA] section 2.11.1)	2-byte in FastTransfer streams, instead of 1-byte as specified in [MS-OXCDATA]. Using **little-endian** byte ordering, "01 00" for "TRUE" and "00 00" for "FALSE".

Property type name	Difference in serialization
PtypString **PtypString8** ([MS-OXCDATA] section 2.11.1)	Serialization MUST be performed, as specified in [MS-OXCDATA]. The server MAY output string values without the terminating nulls.<15> FastTransfer stream readers MUST check that the last 1 (for **PtypString8**) or 2 (for **PtypString**) bytes of a stream are indeed zeros before truncating them.

Note that little-endian byte ordering MUST be used. The data type of simple type elements determine how bytes are serialized on the wire. For example, Int16 value "0x1234" is encoded as "34 12" on the wire.

2.2.4.1.4 Markers

The following table shows the complete list of markers used in FastTransfer streams. The **PidTag** prefix is omitted in the ABNF specified in section 2.2.4.2 to emphasize their difference from properties. Each marker is specified with the **PidTag** prefix in [MS-OXPROPS].

Start/stand-alone marker name and its numeric value		Corresponding end marker , if applicable, and its numeric value	
Folders			
PidTagStartTopFld ([MS-OXPROPS] section 2.1137)	0x40090003	**PidTagEndFolder** ([MS-OXPROPS] section 2.756)	0x400B0003
PidTagStartSubFld ([MS-OXPROPS] section 2.1136)	0x400A0003		
Messages and their parts			
PidTagStartMessage ([MS-OXPROPS] section 2.1134)	0x400C0003	**PidTagEndMessage** ([MS-OXPROPS] section 2.757)	0x400D0003
PidTagStartFAIMsg ([MS-OXPROPS] section 2.1133)	0x40100003		
PidTagStartEmbed ([MS-OXPROPS] section 2.1132)	0x40010003	**PidTagEndEmbed** ([MS-OXPROPS] section 2.755)	0x40020003
PidTagStartRecip ([MS-OXPROPS] section 2.1135)	0x40030003	**PidTagEndToRecip** ([MS-OXPROPS] section 2.758)	0x40040003
PidTagNewAttach ([MS-OXPROPS] section 2.903)	0x40000003	**PidTagEndAttach** ([MS-OXPROPS] section 2.753)	0x400E0003
Synchronization download			
PidTagIncrSyncChg ([MS-OXPROPS] section 2.824)	0x40120003	None.	
PidTagIncrSyncChgPartial ([MS-OXPROPS] section 2.825)	0x407D0003	None.	
PidTagIncrSyncDel ([MS-OXPROPS] section 2.826)	0x40130003	None.	
PidTagIncrSyncEnd ([MS-OXPROPS]	0x40140003	None.	

Start/stand-alone marker name and its numeric value		Corresponding end marker , if applicable, and its numeric value	
section 2.827)			
PidTagIncrSyncRead ([MS-OXPROPS] section 2.833)	0x402F0003	None.	
PidTagIncrSyncStateBegin ([MS-OXPROPS] section 2.834)	0x403A0003	**PidTagIncrSyncStateEnd** ([MS-OXPROPS] section 2.835)	0x403B0003
PidTagIncrSyncProgressMode ([MS-OXPROPS] section 2.831)	0x4074000B	None.	
PidTagIncrSyncProgressPerMsg ([MS-OXPROPS] section 2.832)	0x4075000B	None.	
PidTagIncrSyncMessage ([MS-OXPROPS] section 2.830)	0x40150003	None.	
PidTagIncrSyncGroupInfo ([MS-OXPROPS] section 2.829)	0x407B0102	None.	
Special			
PidTagFXErrorInfo ([MS-OXPROPS] section 2.787)	0x40180003	None.	

The **PidTagStartTopFld** marker signifies the start of data that describes a folder.

The **PidTagStartSubFld** marker signifies the start of serialized data that describes a mailbox subfolder.

The **PidTagEndFolder** marker signifies the end of serialized data that describes a mailbox folder or subfolder.

The **PidTagStartMessage** marker signifies the start of serialized data that describes an e-mail message.

The **PidTagStartFAIMsg** marker signifies the start of serialized data that describes an FAI message.

The **PidTagEndMessage** marker signifies the end of serialized data that describes an e-mail message.

The **PidTagStartEmbed** marker signifies the start of an embedded e-mail message.

The **PidTagEndEmbed** marker signifies the end of an embedded e-mail message.

The **PidTagStartRecip** marker signifies the start of recipient (1) data.

The **PidTagEndToRecip** marker signifies the end of recipient (1) data.

The **PidTagNewAttach** marker signifies the start of an attachment.

The **PidTagEndAttach** marker signifies the end of an attachment.

The **PidTagIncrSyncChg** marker signifies the start of ICS information pertaining to the message.

The **PidTagIncrSyncChgPartial** marker signifies the start of data that describes the property group mapping for properties that have changed in a partial message.

The **PidTagIncrSyncDel** marker signifies the start of deleted message data in the stream.

The **PidTagIncrSyncEnd** marker signifies the end of serialized ICS data.

The **PidTagIncrSyncRead** marker signifies the start of serialized data that describes which messages are to be marked as read or unread.

The **PidTagIncrSyncStateBegin** marker signifies the start of data that describes the synchronization state after ICS finishes.

The **PidTagIncrSyncStateEnd** marker signifies the end of serialized data that describes the synchronization state after ICS finishes.

The **PidTagIncrSyncProgressMode** marker signifies the start of serialized data that describes the size of all the ICS data to be transmitted.

The **PidTagIncrSyncProgressPerMsg** marker signifies the start of the serialized data that describes the size of the next message in the stream.

The **PidTagIncrSyncMessage** marker signifies the start of e-mail data for ICS.

The **PidTagIncrSyncGroupInfo** marker signifies the start of data that describes property group mapping information.

The **PidTagFXErrorInfo** marker signifies the start of error data.

2.2.4.1.5 Meta-Properties

Meta-properties contain information about how to process data, instead of containing data to be processed. Use of meta-properties specified in this section is restricted to specific occasions in FastTransfer streams; therefore, values for these meta-properties are serialized according to FastTransfer stream rules, as specified in section 2.2.4.1.3.

2.2.4.1.5.1 PidTagFXDelProp

A **PtypInteger32** value ([MS-OXCDATA] section 2.11.1) that represents a directive to a client to delete specific subobjects of the object in context. The type of subobjects to delete is determined by the value of the meta-property, which can be any of the property tags specified in section 2.2.1.7.

2.2.4.1.5.2 PidTagEcWarning

A **PtypInteger32** value ([MS-OXCDATA] section 2.11.1) that conveys a warning that occurred when producing output for an element in context.

The following error code requires special processing when passed as a value of the **PidTagEcWarning** meta-property:

Error code name	Description
PartiallyComplete	The client SHOULD verify that properties and subobjects of the object represented by an element in context were output completely.

The complete list of error codes is specified in [MS-OXCDATA] section 2.4.

2.2.4.1.5.3 PidTagNewFXFolder

A **PtypBinary** value ([MS-OXCDATA] section 2.11.1) that provides information about alternative replicas for a public folder in context. Represents a serialized **FolderReplicaInfo** structure.

2.2.4.1.5.4 PidTagIncrSyncGroupId

A **PtypInteger32** value ([MS-OXCDATA] section 2.11.1) that specifies an identifier of a property group mapping. Directs the client to use the specified property group mapping where applicable, until reset with another instance of the **PidTagIncrSyncGroupId** meta-property.

For more details about property groups, see section 3.1.5.4.

2.2.4.1.5.5 PidTagIncrementalSyncMessagePartial

A **PtypInteger32** value ([MS-OXCDATA] section 2.11.1) that specifies an index of a property group within a property group mapping currently in context. Directs a client to treat all forthcoming property values as a part of the specified group, where applicable, until reset with another instance of the **PidTagIncrementalSyncMessagePartial** meta-property.

For more details about property groups, see section 3.1.5.4.

2.2.4.2 Syntactical Structure

The syntactical structure of the FastTransfer adheres to the following guidelines:

- Camel-cased names are nonterminal syntactic elements, as specified in [RFC5234] section 2.3.
- Pascal-cased names without a **PidTag** prefix are markers. Markers are specified in section 2.2.4.1.4 with their **PidTag** prefixes.
- Pascal-cased names with a **PidTag** prefix are **meta-properties** and are also specified in [MS-OXPROPS] section 2.

Note that markers never have a value, and meta-properties, just as regular properties, always have a value when serialized into a FastTransfer stream. Therefore, wherever a marker exists, it is serialized as 4 bytes. Meta-properties, on the other hand, are serialized the same as **propValue** elements.

The syntactical structure of a FastTransfer stream is as follows:

```
root                = contentsSync
                    / hierarchySync
                    / state
                    / folderContent
                    / messageContent
                    / attachmentContent
                    / messageList
                    / topFolder

propValue           = <see lexical structure in 2.2.4.1>
errorInfo           = FXErrorInfo propList
propList            = *propValue

subFolder           = StartSubFld folderContent EndFolder
topFolder           = StartTopFld folderContent EndFolder
folderContent       = propList [PidTagEcWarning]
```

```
                    ( PidTagNewFXFolder / folderMessages )
                    [ PidTagFXDelProp *subFolder ]
folderMessages      = *2( PidTagFXDelProp messageList )
message             = ( StartMessage / StartFAIMsg )
                    messageContent
                    EndMessage
messageChildren     = [ PidTagFXDelProp ] [ *recipient ]
                    [ PidTagFXDelProp ] [ *attachment ]
messageContent      = propList messageChildren
messageList         = 1*( [PidTagEcWarning] message )
recipient           = StartRecip propList EndToRecip

attachment          = NewAttach attachmentContent EndAttach
attachmentContent   = propList [embeddedMessage]
embeddedMessage     = StartEmbed messageContent EndEmbed

contentsSync        = [progressTotal]
                    *( [progressPerMessage] messageChange )
                    [deletions]
                    [readStateChanges]
                    state
                    IncrSyncEnd
hierarchySync       = *folderChange
                    [deletions]
                    state
                    IncrSyncEnd
deletions           = IncrSyncDel propList
folderChange        = IncrSyncChg propList
groupInfo           = IncrSyncGroupInfo propList
messageChange       = messageChangeFull / messageChangePartial
messageChangeFull   = IncrSyncChg messageChangeHeader
                    IncrSyncMessage propList
                    messageChildren
messageChangeHeader = propList
messageChangePartial = [groupInfo] [PidTagIncrSyncGroupId]
                    IncrSyncChgPartial messageChangeHeader
                    *( PidTagIncrementalSyncMessagePartial propList )
                    messageChildren
progressPerMessage  = IncrSyncProgressPerMsg propList
progressTotal       = IncrSyncProgressMode propList
readStateChanges    = IncrSyncRead propList
state               = IncrSyncStateBegin propList IncrSyncStateEnd
```

2.2.4.3 Semantics of Elements

2.2.4.3.1 attachmentContent

The **attachmentContent** element contains the properties and the Embedded Message object of an Attachment object, if present.

Property filters, as specified in section 3.2.5.6, can affect the Attachment object properties in the contained propList.

The following table lists the restrictions that exist on the contained **propList**.

Property name	Restrictions	Comments
PidTagAttachNumber ([MS-OXCMSG] section 2.2.2.6)	Required. Fixed position.	None.
< other properties >	No restrictions.	None.

2.2.4.3.2 contentsSync

The **contentsSync** element contains the result of the content synchronization download operation.

For details about how servers determine the set of differences to be downloaded to clients, see section 3.2.5.1.

2.2.4.3.3 deletions

The **deletions** element contains information about IDs of messaging objects that had been deleted, expired, or moved out of the synchronization scope since the last synchronization, as specified in the initial ICS state. For details about how servers determine the set of IDs to be reported by using this element, see section 3.2.5.1.

Deletions SHOULD NOT be present if SynchronizationFlag NoDeletions was set when configuring the synchronization download operation.

The following restrictions exist on the contained propList :

- MUST contain at least one property.
- MUST adhere to the following restrictions:

Property name	Restrictions	Comments
PidTagIdsetDeleted (section 2.2.1.3.1)	No restrictions	None.
PidTagIdsetNoLongerInScope (section 2.2.1.3.2)	Conditional	MUST NOT be present if **SynchronizationType** equals **Hierarchy**. MUST NOT be present if **SynchronizationFlag IgnoreNoLongerInScope** is set.
PidTagIdsetExpired (section 2.2.1.3.3)	Conditional	MUST NOT be present if **SynchronizationType** equals **Hierarchy**.
< other properties >	*Prohibited*	None.

2.2.4.3.4 errorInfo

The **errorInfo** element provides for out-of-band error reporting and recovery. It is used to provide support for partial completion of the operations by scoping the failures down to the failing object, rather than the entire operation.

The **errorInfo** element can be inserted wherever a lexical structure, specified in section 2.2.4.1, allows a marker or a **propValue**.

This element SHOULD be used if and only if **SendOptions RecoverMode** is set. Note that by the time a server encounters an error that requires failing a download of a messaging object in context, it might have already output some part of the data pertaining to that object in the previous buffer.

Clients MUST support parsing of this element if the client set **RecoverMode** in **SendOptions**.

Whenever a server or a client produces or parses this element, it MUST unwind its producing or parsing stack up to, but not including, the closest element that supports recovery. The current version of the protocol defines two such elements: **contentsSync** and **messageList**. Upon receiving this element, clients can perform additional steps to remove a faulty object from future synchronizations, as specified in [MS-OXCSYNC] section 3.1.5.3.3.

The following table lists the restrictions that exist on the contained **propList**.

Property type name	Restrictions	Comments
[**PtypBinary**] ([MS-OXCDATA] section 2.11.1) 0x00000102	Required Fixed position	Serialized **ExtendedErrorInfo** structure. For more details, see section 2.2.2.9.
< other properties >	*Prohibited*	None.

2.2.4.3.5 folderChange

The **folderChange** element contains a new or changed folder in the hierarchy synchronization.

The contained propList contains the properties of the Folder object, possibly affected by property filters, as specified in section 3.2.5.6, and combined with additional mandatory properties that are required for object identification and conflict detection.

The following table lists the restrictions that exist on the contained **propList**.

Property name	Restrictions	Comments
PidTagParentSourceKey (section 2.2.1.2.6)	Required	None.
PidTagSourceKey (section 2.2.1.2.5)	Required	None.
PidTagLastModificationTime ([MS-OXPROPS] section 2.861)	Required	None.
PidTagChangeKey (section 2.2.1.2.7)	Required	None.
PidTagPredecessorChangeList (section 2.2.1.2.8)	Required	None.
PidTagDisplayName ([MS-OXCFOLD] section 2.3.2.2.3)	Required	None.
PidTagFolderId (section 2.2.1.2.2)	Conditional	MUST be present if and only if **SynchronizationExtraFlag Eid** is set.
PidTagParentFolderId (section 2.2.1.2.4)	Conditional	MUST be present if **SynchronizationFlag NoForeignIdentifiers** is set.
< other properties >	No restrictions	None.

2.2.4.3.6 folderContent

The **folderContent** element contains the content of a folder: its properties, messages, and subfolders.

The propList contains the properties of the Folder object, which are possibly affected by property filters, as specified in section 3.2.5.6.

The following table lists the restrictions that exist on the contained **propList**.

Property name	Restrictions	Comments
PidTagFolderId (section 2.2.1.2.2)	Conditional Fixed position	MUST be present if and only if the folder is not started with **StartTopFld**.
PidTagDisplayName ([MS-OXCFOLD] section 2.3.2.2.3)	Conditional Fixed position	MUST be present if and only if the folder is not started with **StartTopFld**.
PidTagComment ([MS-OXCFOLD] section 2.3.2.2.2)	Conditional Fixed position	MUST be present if and only if the folder is not started with **StartTopFld**.
< other properties >	No restrictions	None.

For more details about the impact of property and subobject filters that are specified when configuring an operation on the content of this element, see section 3.2.5.6.

The **PidTagEcWarning** meta-property (section 2.2.4.1.5.2) MUST be output by the server if the client does not have the permissions necessary to open the folder, to read its contents, view its subfolder structure, or any additional permissions, as specified in section 3.2.5.4.1. The warning is necessary to make it possible for a client to tell this case from an empty folder.

The **PidTagNewFXFolder** meta-property (section 2.2.4.1.5.3) MUST be output instead of message elements when outputting a public folder whose contents do not exist on the server because the content is ghosted. If there is a valid replica on the server and the content has not replicated to the server yet, the content is not included in the synchronization. The server SHOULD NOT include any data following the **PidTagNewFXFolder** meta-property in the buffer. Any data included after this property in the buffer is ignored by the client, which results in a parsing failure when the client attempts to parse the next buffer.

Under conditions specified in section 3.2.5.6, subFolder elements MUST be preceded by a **PidTagFXDelProp** meta-property (section 2.2.4.1.5.1) for the **PidTagContainerHierarchy** property ([MS-OXPROPS] section 2.715).

2.2.4.3.7 folderMessages

The **folderMessages** element contains the messages contained in a folder.

Under conditions specified in section 3.2.5.6, each of these groups MUST be preceded by a **PidTagFXDelProp** meta-property (section 2.2.4.1.5.1) for the corresponding property, **PidTagFolderAssociatedContents** ([MS-OXPROPS] section 2.775) or **PidTagContainerContents** ([MS-OXPROPS] section 2.713), respectively.

2.2.4.3.8 groupInfo

The **groupInfo** element provides a definition for the property group mapping, as specified in section 3.1.5.4. Property group mappings, after they are defined by using the **groupInfo** element, can be referenced with the **PidTagIncrSyncGroupId** meta-property further in the stream by its group ID.

The following table lists the restrictions that exist on the contained **propList**.

Property type name	Restrictions	Comments
[**PtypBinary**] ([MS-OXCDATA] section 2.11.1) 0x00000102	Required Fixed position	Serialized **PropertyGroupInfo** structure. For more details, see section 2.2.2.7.
< other properties >	*Prohibited*	None.

2.2.4.3.9 hierarchySync

The **hierarchySync** element contains the result of the hierarchy synchronization download operation.

For details about how servers determine the set of differences to be downloaded to clients, see section 3.2.5.1.

The parent-child relationship is determined by comparing the **PidTagSourceKey** property (section 2.2.1.2.5) of a prospective parent folder and a **PidTagParentSourceKey** property (section 2.2.1.2.6) of a prospective child folder. The folderChange elements with zero-length **PidTagParentSourceKey** values are children of the root of the synchronization operation.

There MUST be exactly one folderChange element for each descendant folder of the root of the synchronization operation (that is the folder that was passed to the **RopSynchronizationConfigure** ROP) that is new or has been changed since the last synchronization. The folderChange elements for the parent folders MUST be output before any of their child folders.

2.2.4.3.10 message

The message element represents a Message object.

The type of the starting marker to use depends on whether the message is a normal message or an FAI message. Normal messages use the **StartMessage** marker; FAI messages use the **StartFAIMsg** marker.

2.2.4.3.11 messageChange

The **messageChange** element represents a change to a Message object.

A server MUST use **messageChangeFull**, instead of **messageChangePartial**, if any of the following are true:

- SendOptions PartialItem flag was not set.
- The **MID** value ([MS-OXCDATA] section 2.2.1.2) of the message to be output is not in the **PidTagIdsetGiven** property (section 2.2.1.1.1) from the initial ICS state.
- The message is an FAI message.

- The message is a conflicting version contained in a conflict resolve message. For more details, see section 3.1.4.1.2.1.

Otherwise, it is up to the server to determine the most efficient way to communicate the message change on a case-by-case basis.

2.2.4.3.12 messageChildren

The **messageChildren** element represents child objects of the Message objects: **Recipient objects** and Attachment objects.

For more details about the impact of property and subobject filters that are specified when configuring an operation on the content of this element, see section 3.2.5.6.

Under the conditions specified in section 3.2.5.6, recipient (1) and attachment elements MUST be preceded by a **PidTagFXDelProp** meta-property (section 2.2.4.1.5.1) for the **PidTagMessageRecipients** ([MS-OXPROPS] section 2.892) and **PidTagMessageAttachments** ([MS-OXPROPS] section 2.882) properties, respectively.

2.2.4.3.13 messageChangeFull

The **messageChangeFull** element contains the complete content of a new or changed message: the message properties, the recipients (1), and the attachments.

Property filters, as specified in section 3.2.5.6, can affect the Message object properties in the contained **propList**.

2.2.4.3.14 messageChangeHeader

The **messageChangeHeader** element contains a fixed set of information about the message change that follows this element in the FastTransfer stream. The information in the header is sufficient for message identification and conflict detection.

The following table lists the restrictions that exist on the contained **propList**.

Property name	Restrictions	Comments
PidTagSourceKey (section 2.2.1.2.5)	Required Fixed position	None.
PidTagLastModificationTime ([MS-OXPROPS] section 2.861)	Required Fixed position	None.
PidTagChangeKey (section 2.2.1.2.7)	Required Fixed position	None.
PidTagPredecessorChangeList (section 2.2.1.2.8)	Required Fixed position	None.
PidTagAssociated (section 2.2.1.4)	Required Fixed position	None.

Property name	Restrictions	Comments
PidTagMid (section 2.2.1.2.1)	Conditional	MUST be present if and only if **SynchronizationExtraFlag Eid** is set.
PidTagMessageSize (section 2.2.1.5)	Conditional	MUST be present if and only if **SynchronizationExtraFlag MessageSize** is set.
PidTagChangeNumber (section 2.2.1.2.3)	Conditional	MUST be present if and only if **SynchronizationExtraFlag Cn** is set.
< *other properties* >	*Prohibited*	None.

2.2.4.3.15 messageChangePartial

The **messageChangePartial** element<16> represents the difference in message content since the last download, as identified by the initial ICS state. Changes to a message are output based on the granularity of the property group, as specified in section 3.1.5.4. The last encountered **PidTagIncrSyncGroupId** meta-property determines which property group mapping MUST be used.

Clients MUST treat every contained propList element as the complete content of a property group denoted by the **PidTagIncrementalSyncMessagePartial** meta-property that preceded it. That is, all properties missing from a **propList**, but defined for this group in the corresponding property group mapping, MUST be deleted from the Message object.

The following table lists the restrictions that exist on the contained **propList** elements.

Property type name	Restrictions	Comments
[**PtypInteger32**] ([MS-OXCDATA] section 2.11.1) 0x00000003	Conditional	MUST be present if and only if a property group is empty, but was still marked as changed since the last download. Value MUST be "0". MUST be ignored by clients.
< *other properties* >	No restrictions	None.

2.2.4.3.16 messageContent

The **messageContent** element represents the content of a message: its properties, the recipients (1), and the attachments.

Property filters, as specified in section 3.2.5.6, can affect the Message object properties in the contained **propList**.

Property name	Restrictions	Comments
PidTagMid (section 2.2.1.2.1)	Required Fixed position	Clients MUST ignore the value of this property for Embedded Message objects.
< *other properties* >	No restrictions	None.

2.2.4.3.17 messageList

The **messageList** element contains a list of messages, which is determined by the scope of the operation.

For each message in the **messageList**, the server SHOULD output **PidTagEcWarning** meta-property (section 2.2.4.1.5.2) if a client does not have the permissions necessary to access it, as specified in section 3.2.5.4.1. The warning is necessary to make it possible for a client to tell this case from a missing message.

2.2.4.3.18 progressPerMessage

The **progressPerMessage** element contains data that describes the approximate size of message change data that follows.

MUST be present if and only if the **progessTotal** element was output within the same ancestor **contentsSync** element.

MUST NOT be present if **SynchronizationFlag Progress** was not set when configuring the synchronization download operation.

The following table lists the restrictions that exist on the contained **propList**.

Property type name	Restrictions	Comments
[**PtypInteger32**] ([MS-OXCDATA] section 2.11.1) 0x00000003	Required Fixed position	Size of the message to be follow. Servers can supply the same value as the **PidTagMessageSize** property (section 2.2.1.5) in **messageChangeHeader**, or use a different approximation.
[**PtypBoolean**] ([MS-OXCDATA] section 2.11.1) 0x0000000B	Required Fixed position	"TRUE" (0x0001) if the Message object that follows is FAI; otherwise, "FALSE" (0x0000). For more details about the serialization of **PtypBoolean** values in FastTransfer streams, see section 2.2.4.1.3.
< other properties >	*Prohibited*	None.

2.2.4.3.19 progressTotal

The **progressTotal** element contains data that describes the approximate size of all the **messageChange** elements that will follow in this stream. This element can be used by clients to display progress information. Servers can use a sum of message sizes (**PidTagMessageSize** property (section 2.2.1.5)) for all messages in which changes will be downloaded in the current operation, or servers can use a different approximation.

Note that this method of reporting progress is provided in addition to what is available in the the **RopFastTransferSourceGetBuffer** ROP response. This method of reporting is supposed to reflect the amount of work more precisely, as it is based on message sizes, rather than object count.

This element MUST be present if SynchronizationFlag Progress was set when configuring the synchronization download operation, and a server supports progress reporting.

This element MUST NOT be present if SynchronizationFlag Progress was not set when configuring the synchronization download operation.

The following table lists the restrictions that exist on the contained propList.

Property type name	Restrictions	Comments
[**PtypBinary**] ([MS-OXCDATA] section 2.11.1) 0x00000102	Required Fixed position	Serialized **ProgressInformation** structure. For more details, see section 2.2.2.6.
< other properties >	*Prohibited*	None.

2.2.4.3.20 propList

The **propList** elements MUST NOT contain **propValue** elements for **meta-properties**. All instances in which meta-properties, as specified in section 2.2.4.1.5, can be encountered in a document are mentioned explicitly in the syntax ABNF.

Syntactic elements that contain a **propList** can express restrictions on a set of properties and/or the position of properties within a list by using property list restriction table syntax, as specified in section 2.2.

Properties that contain an error (have the **PtypErrorCode** type, as specified in [MS-OXCDATA] section 2.11.1) instead of an actual value MUST be omitted from the **propList**.

2.2.4.3.21 propValue

The <propValue> element represents identification information and the value of the property.

Note that the protocol imposes no limit on the size of data that can be encoded using this element, unlike the response buffers of the **RopQueryRows** ROP and the **RopGetPropertiesSpecific** ROP. Clients and servers MUST be capable of accepting large amounts of data and MUST fail the operation if the size of data crosses the threshold imposed by an implementation, rather than truncating the data.

2.2.4.3.22 readStateChanges

The **readStateChanges** element contains information about **MID** values ([MS-OXCDATA] section 2.2.1.2) of Message objects that had their read state changed since the last synchronization, as specified by the initial ICS state. For details about how servers determine the set of IDs to be reported by using this element, see section 3.2.5.1.

This element SHOULD NOT be present if **SynchronizationFlag ReadState** was not set when configuring the synchronization download operation.

The following restrictions exist on the contained **propList**:

- MUST contain at least one property.
- MUST adhere to the following restrictions:

Property name	Restrictions	Comments
PidTagIdsetRead (section 2.2.1.3.4)	No restrictions	None.
PidTagIdsetUnread (section 2.2.1.3.5)	No restrictions	None.
< other properties >	*Prohibited*	None.

2.2.4.3.23 recipient

The recipient element represents a Recipient object, which is a subobject of the Message object.

The **propList** child element contains the properties of the Recipient object.

The following table lists the restrictions that exist on the contained **propList**.

Property name	Restrictions	Comments
PidTagRowid ([MS-OXPROPS] section 2.1035)	Required Fixed position	None.
< other properties >	No restrictions	None.

2.2.4.3.24 root

The **root** element contains the root element of FastTransfer streams.

Producers of the FastTransfer stream MUST choose a contained element to generate depending on the Bulk Data Transfer operation in effect. For more details, see the mapping specified in section 3.1.5.8 and section 2.2.3.1.2.1.1.

2.2.4.3.25 state

The **state** element contains the **final ICS state** of the synchronization download operation. For details about how servers construct the final ICS state, see sections 3.1.5.1 and 3.2.5.1.

The following table lists the restrictions that exist on the contained **propList**.

Property name	Restrictions	Comments
PidTagIdsetGiven (section 2.2.1.1.1)	No restrictions	None.
PidTagCnsetSeen (section 2.2.1.1.2)	No restrictions	None.
PidTagCnsetSeenFAI (section 2.2.1.1.3)	Conditional	MUST NOT be present if **SynchronizationType** equals **Hierarchy**.
PidTagCnsetRead (section 2.2.1.1.4)	Conditional	MUST NOT be present if **SynchronizationType** equals **Hierarchy**.
< other properties >	*Prohibited*	None.

3 Protocol Details

3.1 Common Details

The protocol details in this section contain formulas operating on sets of elements, which include the operators and special identifiers listed in the following table.

Operator or special identifier	Example	Definition
∪	A ∪ B	Union of two sets. Every element in the resulting set belongs to either A, or B, or both.
∩	A ∩ B	Intersection of two sets. Every element in the resulting set belongs to both A and B.
{ }	{A1,..., An}	A set consisting of elements A1 through An.
⊆ ⊇	B ⊆ A A ⊇ B	B is a subset of or equal to A: every element of B is also an element of A.
+=	Set += element	Instructs to include an element into a set. The Set is assigned to Set {element}.
ø	A = ø	Empty set: a set that contains no elements. Set A is asserted to be an empty set, it has no elements.
\	C =A \ B	Relative compliment: the elements belonging to A that are not in B. Set C is the relative compliment of sets A and B.

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 following common abstract object types are defined in this document:

Global

Mailbox

Messaging Object

ICS State

3.1.1.1 Global

There are no global parameters defined as common to both client and server.

3.1.1.2 Per Mailbox

Mailboxes are represented by the **Mailbox** abstract data type. The following abstract data elements are maintained for each **Mailbox**:

Mailbox.MessagingObject: A Folder object, Attachment object, or Message object only.

3.1.1.3 Per Messaging Object

Messaging objects are represented by the **MessagingObject** abstract data type. The following abstract data elements are maintained for each **MessagingObject**:

MessagingObject.Mid: An identifier for a **Mailbox.MessagingObject** that is a Message or Attachment object, as specified in section 2.2.1.2.1.

MessagingObject.FolderId: An identifier for a **Mailbox.MessagingObject** that is a Folder object, as specified in section 2.2.1.2.2.

MessagingObject.ParentFolderId: An identifier for a **Mailbox.MessagingObject** that is a Folder object containing another Folder object, as specified in section 2.2.1.2.4.

MessagingObject.ChangeNumber: An identifier for a version of a **Mailbox.MessagingObject**, as specified in section 2.2.1.2.3.

3.1.1.4 Per ICS State

ICS states are represented by the **ICSState** abstract object type. Each **ICSState** represents the state of either a content synchronization or a hierarchy synchronization operation. The following abstract data elements are maintained for each **ICSState**:

ICSState.State: A state that identifies the **Mailbox.MessagingObjects** that have been communicated to the client at a particular point in time. The following **ICSState.State** values identify the point in time the **ICSState.State** represents:

- **Initial**. The ICS state provided by the client at the beginning of the ICS operation. The server compares the values of the initial ICS state properties to its version, and downloads the differences.
- **Checkpoint**. The ICS state provided by the server during the ICS operation.
- **Final**. The ICS state provided by the server at the end of the ICS operation.

ICSState.SeenNormal: Contains a set of **Mailbox.MessagingObject.ChangeNumber** values that identify changes to normal messages that have been communicated to the client.

ICSState.SeenFAI: Contains a set of **Mailbox.MessagingObject.ChangeNumber** values that identify changes to FAI messages that have been communicated to the client.

ICSState.Read: Contains a set of **Mailbox.MessagingObject.ChangeNumber** values that identify the read state changes of messages that have been communicated to the client.

ICSState.IdsetGiven: Contains a set of **Mailbox.MessagingObject.Mid** or **Mailbox.MessagingObject.FolderId** values that exist on the client.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Conflict Handling

The properties that are associated with a message or a folder can be modified by the server or client at any time. Synchronizing these changes can result in conflicts in which a server or a client has to decide which set of message properties or folder properties to use: the local copy, or the copy being replicated.

This specification does not mandate that clients implement any **conflict handling**. However, if clients do implement conflict handling, their conflict handling logic MUST be compatible with the one mandated for servers, as specified in this section, to ensure the consistency of user experience regardless of the protocol role performing the conflict handling. When referring to synchronization in this specification, both download and upload are considered, unless specified otherwise.

3.1.4.1.1 Detection

Servers MUST implement conflict detection using an algorithm compatible with the one described in this section.

Servers MUST perform conflict detection on ICS uploads for versions of messaging objects stored in a server replica and passed by the client through the **RopSynchronizationImport*** ROPs.

Conflict detection is performed by examining the **PidTagPredecessorChangeList** properties (section 2.2.1.2.8) for objects that have the same value for the **PidTagSourceKey** property (section 2.2.1.2.5).

Clients can perform conflict detection during ICS download for versions of objects stored in a local replica and passed by the server in a FastTransfer stream.

To illustrate the use of **PCLs** in conflict detection, the following algorithm uses sample PCLs (PCLA and PCLB) to detect a conflict between two versions of the same messaging object.

Conflict Detection Algorithm

PCLA includes PCLB if and only if for every **XID**, as specified in section 2.2.2.2, in PCLB there is an **XID** in PCLA that has the same NamespaceGuid and same or greater LocalId part. The notation PCLA ≥ PCLB will be used if PCLA includes or is equal to PCLB.

If a change to a messaging object is being synchronized from replica A to replica B, use the following statements to identify the conflict and the version to replicate:

1. If PCLA includes PCLB, then the version from replica A is newer and replaces the version in replica B.

2. If PCLB includes or is equal to PCLA, then the version from replica A is older, and is ignored. The version in replica B remains intact.

3. If neither 1 nor 2 is true, then versions from replicas A and B are in conflict.

Servers can create and implement additional conflict detection mechanisms, as long as PCLs for object versions that do and do not conflict adhere to these criteria.

The following figure shows how to detect a synchronization conflict when comparing the **PidTagPredecessorChangeList** property (section 2.2.1.2.8).

The following figure simplifies the contents of the PCL to focus on the comparison. Each **CN** structure, as specified in section 2.2.2.1, within the PCL in the figure is represented by a letter (A for server changes, and B for client changes) and an increasing number.

The server-side PCL is missing a B2 change, while the client side PCL is missing A2; therefore, each has changes the other has not seen, and thus these modifications are conflicting.

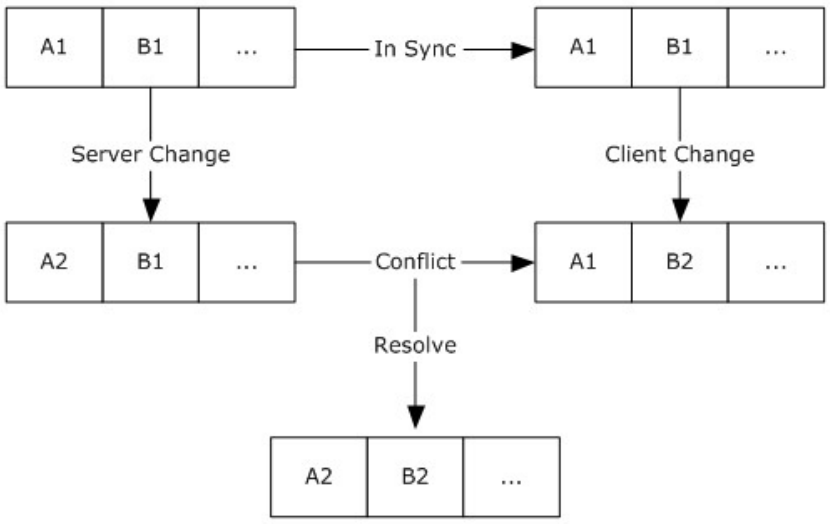


Figure 1: Conflict details

The following sections describe the details of synchronization when detecting conflicting changes.

3.1.4.1.2 Resolution

At a minimum, servers MUST implement conflict resolution to the extent specified in this section. Servers can implement additional resolution algorithms. Any additional resolution algorithms MUST NOT result in the creation of conflict resolve messages, as specified in section 3.1.4.1.2.1.

A version that results from conflict resolution MUST have a PCL that makes it a successor of all conflicting versions. To achieve that, protocol roles SHOULD assign the successor a PCL created by merging the PCLs of all conflicting versions.

Version X is a successor of versions A and B if and only if the conflict detection algorithm specified in section 3.1.4.1.1 would determine that X is not in conflict and is newer than both A and B.

PCLX is a **merge** of PCLA and PCLB if and only if all of the following statements are true:

$PCLX \subseteq (PCLA \cup PCLB)$

$PCLX \geq PCLA$

$PCLX \geq PCLB$

3.1.4.1.2.1 Conflict Resolve Message

A conflict resolve message provides a way to encapsulate conflicting versions of a Message object into a single Message object, by storing all the versions of the Message object as individual attachments to the new Message object and choosing a temporary winning message and copying it as the message contents. The contents of the conflict resolve message include all properties and subobjects of the winning version; therefore the conflict resolve message can be used in place of the winning version whenever needed. The winner MUST be determined by the last writer wins algorithm, as specified in section 3.1.4.1.2.2. Because the conflict resolve message is a successor of all the conflicting versions it represents, its PCL MUST be the merge of the PCLs of the conflicting versions.

Conflict resolve messages MUST NOT be synchronized as Message objects. Instead, each attachment that represents a version in conflict MUST be synchronized as a separate Message object. This allows the other protocol role to re-resolve the conflict during synchronization, while considering all (possibly, more than two) conflicting versions. The other protocol role MUST then generate a new message that matches the structure of the conflict message.

A conflict resolve message MUST contain the **msInConflict** flag in the **PidTagMessageStatus** property ([MS-OXCMSG] section 2.2.1.8). Each attachment that represents an alternate replica MUST have the value of **PidTagInConflict** set to "TRUE". This allows them to be distinguished from other "regular" attachments on the message.

The client and server MUST generate a conflict-resolve message when detecting a conflict against the current version of a message in the replica during synchronization. It is important to understand that it is possible that the current version of the message in the local replica was transmitted during the current synchronization operation. This will happen when the conflict already exists on the server before any of the conflicting messages were downloaded to the local replica.

3.1.4.1.2.2 Last Writer Wins Algorithm

The last writer wins algorithm uses the **PidTagLastModificationTime** ([MS-OXPROPS] section 2.861) property to determine the winning version of the folder or message, as specified in the following steps:

1. The version with the most recent **PidTagLastModificationTime** wins.

2. For messages, if the **PidTagLastModificationTime** value is equal on both objects, the tie-breaking winner is determined by comparing byte-to-byte values of the **NamespaceGuid** field for **XIDs**, as specified in section 2.2.2.2, in the **PidTagChangeKey** properties (section 2.2.1.2.7). The message with the larger **NamespaceGuid** field wins. For folders, if the **PidTagLastModificationTime** value is equal on both objects, the server version is kept.

3. If the byte-to-byte comparison in step 2 determines that the **NamespaceGuid** fields are equal, the version being imported wins.

The last writer wins algorithm MUST be used for conflicts detected during hierarchy synchronization and content synchronization operations on normal messages (unless specified otherwise in the **PidTagResolveMethod** property (section 2.2.1.6) set on the folder) as well as FAI messages, and folders.

3.1.4.1.3 Reporting

Conflict reporting, if deemed necessary by the value of the **PidTagResolveMethod** property (section 2.2.1.6) of the folder, SHOULD be done through a combination of the following methods:

1. Failing the ROP that detected the conflict.

2. Creating a conflict resolve message.

3. Creating a conflict notification message, as specified in section 3.1.4.1.3.1.

Servers MUST implement conflict reporting by failing ROPs and creating conflict resolve messages. Servers MAY implement other means of conflict reporting.

The use of the conflict resolve message combines semi-automatic conflict resolution with conflict reporting: the message has all properties of the winning version, while at the same time it contains all conflicting versions as attachments, which clients can use to offer manual conflict resolution.

Determining whether to perform conflict reporting, and what method of conflict reporting to use, is dependent on the operation that triggered the conflict detection, as specified in section 3.1.4.1.1, and on the value of the **PidTagResolveMethod** property on the folder, whose values are specified in section 2.2.1.6.

This controls whether the **RopSynchronizationImportMessageChange** ROP is required to perform conflict reporting by failing the ROP or by creating a conflict notification message. However, the **RopSynchronizationImportHierarchyChange** ROP MUST detect and resolve, and SHOULD report, possible conflicts by using a conflict notification message.

3.1.4.1.3.1 Conflict Notification Message

A conflict notification message is a special message used to notify the owner of a public folder that a conflict was resolved. This message is identified by setting the value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) to "IPM.Conflict.Message" which is used to notify a user that a conflict resolve note has been created. This notification MUST NOT be generated for public folder conflicts if the RESOLVE_NO_CONFLICT_NOTIFICATION flag is present in the **PidTagResolveMethod** property (section 2.2.1.6) for the public folder, as specified in [MS-OXCSTOR].

A conflict notification message MUST include the properties in the following table, as specified in [MS-OXCMSG].

Property	Description
PidTagSenderName ([MS-OXOMSG] section 2.2.1.43)	Name of the folder that contains the conflict resolve message.
PidTagOriginalSubject ([MS-OXOMSG] section 2.2.2.16)	Original subject of the message.
PidTagConflictEntryId ([MS-OXPROPS] section 2.710)	**EntryID** of the conflict resolve message.

3.1.5 Message Processing Events and Sequencing Rules

ROPs discussed in this document are synchronous and MUST be executed in the order outlined for each operation specified in sections 3.2.5 and section 3.3.5 and their subsections. Otherwise, the client and server behavior remains undefined.

3.1.5.1 Isolating Download and Upload Operations

Upload and download operations are not always isolated transactions. Upload and download operations can be affected by other operations on messaging objects.

To counteract the lack of transaction isolation between ICS download operations and the rest of operations that occur on messaging objects at the same time, servers MUST guarantee that the final ICS state does not reflect the state of the server replica at the end of the operation, but instead reflects the actual differences downloaded to a client, combined with the initial ICS state.

3.1.5.2 Managing ICS State Properties

By using the ICS state properties specified in section 2.2.1.1, only differences that are relevant to a client are downloaded and the same information is only downloaded once. The ICS state is produced by the server, optionally modified by the client, and persisted exclusively on the client. The client passes the ICS state to the server immediately after configuring a synchronization context for download or upload. The server uses the ICS state and the synchronization scope, as defined during initialization of the synchronization download context, to determine the set of differences to download to the client. At the end of the synchronization operation, the client is given a new ICS state, commonly referred to as the final ICS state.

ICS state properties are not persisted on the server and are only present as data in the FastTransfer stream and in the fields of ROPs that support synchronization. The server uses the synchronization scope and ICS state to determine what differences to download to the client. For more server-specific details, see section 3.2.5.1. Ordinarily, the server modifies the ICS state properties and sends them back to the client in the FastTransfer stream or ROP responses. Another method of sending state information back to the client is checkpointing, as specified in section 3.3.5.3.

Note that for the purposes of reducing the wire size of the ICS state by enabling compacting of regions, as specified in section 3.1.5.6, and optimizing for performance of determining a set of differences to be downloaded to clients, servers can include extra IDs in **IDSETs** that represent CNSETs, as specified in section 2.2.2.4, as long as that will never affect the sets of differences that are downloaded to clients.

During the first synchronization of a synchronization scope, a client MUST send the relevant ICS state properties as zero-length byte arrays. The server assumes that the ICS state properties are zero-length byte arrays if a client fails to send them when setting up a content synchronization download operation. It is recommended that clients always send all ICS state properties that are relevant to a selected synchronization mode, defaulting them to zero-length byte arrays.

3.1.5.2.1 Sending and Receiving the PidTagIdsetGiven ICS State Property

The property tag for this property suggests that it is of type **PtypInteger32** ([MS-OXCDATA] section 2.11.1), but the data MUST be handled as **PtypBinary** ([MS-OXCDATA] section 2.11.1) data by both clients and servers. Clients and servers SHOULD send the **PidTagIdsetGiven** property (section 2.2.1.1.1) with a property tag that defines it as **PtypInteger32**; however, servers SHOULD accept this property when the property tag identifies it as **PtypInteger32** or **PtypBinary**.

This property is ignored for synchronization upload operations and is not downloaded back to the client in the final ICS state obtained for them through the **RopSynchronizationGetTransferState** ROP. Clients SHOULD remove this property before uploading the initial ICS state on synchronization upload contexts and clients MUST merge this property back in when receiving the final ICS state from the server. However, if the client does not remove this property before uploading the initial ICS state, there is no server impact. Clients MUST add IDs of messaging objects created in or originating

from a local replica to this property by using a process called checkpointing, as specified in section 3.3.5.3.

3.1.5.3 Identifying Objects and Maintaining Change Numbers

On creation, objects in the mailbox are assigned internal identifiers (2), commonly known as **FID** values ([MS-OXCDATA] section 2.2.1.1) for folders and **MID** values ([MS-OXCDATA] section 2.2.1.2) for messages. After internal identifiers (2) are assigned to an object, they MUST never be reused, even if the object it was first assigned to no longer exists. Copying of messaging objects within a mailbox or moving messages between folders of the same mailbox translates into creation of new messaging objects and therefore, new internal identifiers (2) MUST be assigned to new copies. All other observed behavior is an implementation detail, and not a part of the protocol, and therefore MUST NOT be relied upon.

In most cases, the server is responsible for assigning internal identifiers (2) to mailbox objects, which usually happens during execution of ROPs, such as **RopSaveChangesMessage** and **RopCopyTo**, or while processing events not controlled by the client (such as Message object delivery).

Messaging objects also maintain a change number by using the **CN** structure, as specified in section 2.2.2.1, which identifies a version of an object and adheres to the same rules as internal identifiers (2) for messaging objects. When a new object is created, it is assigned a **CN** value. A new **CN** value is assigned to a messaging object each time it is modified. For messages, in addition to a **CN** value for the entire message, there are additional mechanisms for tracking changes to their elements, such as the read state, as specified in section 3.2.5.3, and properties and subobjects arranged into groups, as specified in section 3.1.5.4.

A protocol role that generates internal identifiers (2) for messaging objects and changes MUST ensure that the GLOBCNT portions of the internal identifiers (2) that share the same REPLGUID, as specified in the **XID** structure in section 2.2.2.2, only increase with time, when compared byte to byte.

Whenever a change number is changed on a messaging object as the result of the direct modification of the object in a replica, as opposed to a synchronization, its Predecessor Change List (PCL) MUST be merged with the **XID** that represents the new change number.

Clients that use ICS upload to synchronize their local replica with a server replica MUST assign identifiers to client-originated objects in a local replica by using one of the mechanisms specified in section 3.3.5.1.1. Clients MUST generate foreign identifiers, as specified in section 3.3.5.1.3, to identify client-side changes to objects that they export through ICS upload.

Upon successful import of a new or changed object using ICS upload, the server MUST do the following when receiving the **RopSaveChangesMessage** ROP:

- Assign the object a new internal change number (**PidTagChangeNumber** property (section 2.2.1.2.3)).
- This is necessary because the server MUST be able to represent the imported version in the **PidTagCnsetSeen** (section 2.2.1.1.2) or **PidTagCnsetSeenFAI** (section 2.2.1.1.3) properties, and these properties cannot operate on **foreign identifiers** for change numbers that a client passes.
- Assign the object an internal identifier (2), such as a **PidTagMid** value (section 2.2.1.2.1) or a **PidTagFolderId** value (section 2.2.1.2.2), based on the kind of external identifier that was passed for the objects identification by the client if and only if the object is new.

- If the external identifier is a **GID** value ([MS-OXCDATA] section 2.2.1.3), the server MUST convert it to a short-term internal identifier (2) and assign it to an imported object.
- Assign the object the given **PidTagChangeKey** property value (section 2.2.1.2.7) and **PidTagPredecessorChangeList** (section 2.2.1.2.8) that equals PCL {**PidTagChangeKey**}.

If the import of the object triggered detection of a conflict, the server MUST follow the previous steps for a version of the object resulting from the conflict resolution. For details about handling conflict, see section 3.1.4.1.

Foreign identifiers supplied by clients for change identification (such as the **PidTagChangeKey** property) are replaced whenever their corresponding internal identifiers (2) change. Examples are provided in the following table. The table uses the following notation:

- Equals sign (=) to specify that a property is set to the value specified and uses the equals and p. For example, **PidTagSourceKey** = GID(ID1) means that the **PidTagSourceKey** property is set to the value of the initial **global identifier**.
- Plus sign followed by equals sign (+=) to specify that the value has been incremented. For example, an initial change number of 1 (CN1) increments to 2 (CN2) as changes are made to the message or folder.

Sequence of client action	Updates made on the server
RopSynchronizationImportMessageChange ROP for a new message: ▪ **PidTagSourceKey** = GID(ID1) ▪ **PidTagChangeKey** = XCN1 Client checkpoints the stored initial ICS state: ▪ **PidTagIdsetGiven** += ID2	▪ **PidTagSourceKey** = GID(ID1) ▪ **PidTagMid**= ID1 ▪ **PidTagChangeKey** = XCN1 ▪ **PidTagChangeNumber** = CN2 ▪ Final ICS state: **PidTagCnsetSeen** += CN2
RopSynchronizationImportMessageChange ROP ▪ **PidTagSourceKey** = GID(ID1) ▪ **PidTagChangeKey** = XCN3	▪ **PidTagChangeKey** = XCN3 ▪ **PidTagChangeNumber** = CN4 ▪ Final ICS state: **PidTagCnsetSeen** += CN4
ICS download of contents	▪ **PidTagSourceKey** = GID(ID1) ▪ **PidTagMid** = ID1 ▪ **PidTagChangeKey** = XCN3 ▪ **PidTagChangeNumber** = CN4
RopOpenMessage ROP **RopSetProperties** ROP **RopSaveChangesMessage** ROP	▪ **PidTagChangeNumber** = CN5

Sequence of client action	Updates made on the server
ICS download	▪ Changes to a message: ▪ **PidTagSourceKey** = GID(ID1) ▪ **PidTagMid** = ID1 ▪ **PidTagChangeKey** = GID(CN5) ▪ **PidTagChangeNumber** = CN5 ▪ Final ICS state: **PidTagCnsetSeen** += CN5
RopSynchronizationImportMessageMove	▪ Message is hard deleted in the source folder A. ▪ A copy of the message is created in destination folder B with: ▪ **PidTagMid** = ID2 ▪ **PidTagChangeNumber** = CN6
ICS download of contents for folder A	▪ Deletions: ID1 ▪ Final ICS state: **PidTagIdsetGiven** -= ID1
ICS download of contents for folder B	▪ New message: ▪ **PidTagSourceKey** = GID(ID2) ▪ **PidTagMid** = ID2 ▪ **PidTagChangeKey** = GID(CN6) ▪ **PidTagChangeNumber** = CN6 ▪ Final ICS state: ▪ **PidTagIdsetGiven** -= ID2 ▪ **PidTagCnsetSeen** += CN6
RopSynchronizationImportMessageChange ▪ **PidTagSourceKey** = GID(ID2) ▪ **PidTagChangeKey** = XCN7	▪ **PidTagChangeKey** = XCN7 ▪ **PidTagChangeNumber** = CN8

3.1.5.4 Working with Property Groups and Partial Changes

Servers that are implemented to support<17> partial message change synchronization MUST either use a mechanism described in this section, or use an alternative mechanism that localizes changes to a message to a set of properties and subobjects, which can be unambiguously expressed by using the **messageChangePartial** element, as specified in section 2.2.4.3.15, of the FastTransfer stream. Servers that are not implemented to support partial message change synchronization ignore the **SendOptions PartialItem** flag, as specified in section 2.2.3.1.1.2, and download the item as a full item by using the **messageChangeFull** element, as specified in section 2.2.4.3.13, of the FastTransfer stream.

ICS is optimized for reporting partial changes to messages on a property group basis. The simplest approach for servers providing that information is to track changes made to groups of properties. A group is considered changed if any of the properties in the group are modified. It is up to the server to define a property group mapping - how properties are distributed into groups. ICS offers a way to communicate property group mapping information per-message, so every message can use its own property group mapping. However, to minimize overhead, it is recommended that the number of different mappings is kept to a minimum.

For example, a change to any single attachment property would mean that all the properties in the attachment property group are updated during ICS. Likewise, a change to any one body property would mean that all the properties in the body property group are updated during the next synchronization.

To track changes to property groups on a message, servers SHOULD keep change numbers for each property group, and assign a new change number to both the group and the message whenever a change is made to a property that belongs to the group. Note that marking a message as read or unread is the most common type of message modification, and there is a specific mechanism to support just that change, as specified in section 3.2.5.3.

How properties are organized into property groups determines their property group mapping. One message in a mailbox can have a different mapping than another message, which means that the properties in group N on one message can be different than the properties in group N in another message. Property group mappings do not change frequently, but they do change with server upgrades. When a message is modified and the default mapping has changed after an upgrade, the property group mapping of the message is updated.

3.1.5.5 Serializing an IDSET

When an **IDSET** has to be transmitted from a client to a server or from a server to a client, it has to be serialized. This section specifies details about how to serialize an **IDSET**.

3.1.5.5.1 Formatted IDSET

Before serialization, the contents of an **IDSET** have to be arranged in such a way as to allow it to be properly encoded. The ID values MUST be arranged by REPLID and all IDs for each REPLID MUST be reduced into a **GLOBSET** of GLOBCNT values. Each **GLOBSET** MUST be arranged from lowest to highest GLOBCNT where all duplicate GLOBCNT values are removed. The remaining GLOBCNT values MUST be grouped into consecutive ranges with a low GLOBCNT value and a high GLOBCNT value. If a GLOBCNT value is disjoint it MUST be made into a singleton range with the low and high GLOBCNT values being the same. The following figure shows what a properly formatted **IDSET** looks like for serialization.

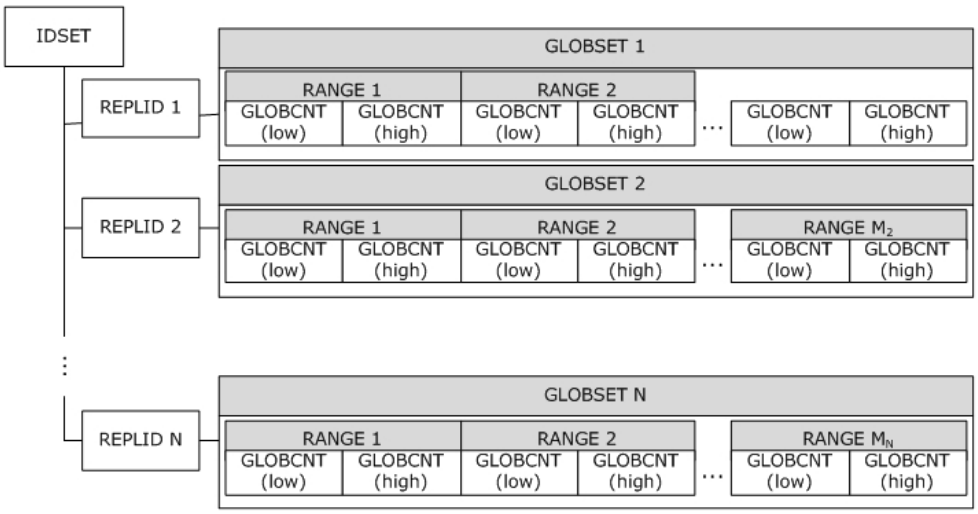


Figure 2: Formatted IDSET

3.1.5.5.2 IDSET Serialization

There are two different formats in which a serialized **IDSET** can exist on the wire. The only difference is how the REPLID value is represented in the serialization buffer. The first format contains the REPLID value followed by the **GLOBSET** data and is used by the following properties: **PidTagIdsetDeleted** (section 2.2.1.3.1), **PidTagIdsetNoLongerInScope** (section 2.2.1.3.2), **PidTagIdsetExpired** (section 2.2.1.3.3), **PidTagIdsetRead** (section 2.2.1.3.4), and **PidTagIdsetUnread** (section 2.2.1.3.5). The second format contains, instead of the REPLID, the REPLGUID that is associated with the REPLID, followed by the **GLOBSET** data, and it is used by the following properties: **PidTagIdsetGiven** (section 2.2.1.1.1), **PidTagCnsetSeen** (section 2.2.1.1.2), **PidTagCnsetSeenFAI** (section 2.2.1.1.3), **PidTagCnsetRead** (section 2.2.1.1.4). No information contained in the serialized buffer identifies which format is being used. The context in which the serialized **IDSET** is being used on the wire dictates which format MUST be used: if an **IDSET** was persisted or is intended to be persisted across sessions, such as when it represents a portion of an ICS state, as specified in section 2.2.1.1, it MUST be transmitted in the REPLGUID-based form. If it's only a part of a transient set of data, like IDs of items that were deleted since the last synchronization, as specified in section 2.2.1.3.1, it MUST be transmitted in a REPLID-based form. Sections 3.1.5.5.3 through section 3.1.5.5.3.2.5 specify the layout of both formats on the wire. REPLID-based format can be converted to REPLGUID-based format by using mapping operations, as specified in [MS-OXCSTOR].

For more details about the format of each serialized **IDSET**, see section 2.2.2.4.

3.1.5.5.3 GLOBSET Serialization

IDSET serialization requires each **GLOBSET** within the **IDSET** to be serialized. The GLOBCNT ranges within the **GLOBSET** are serialized by using special encoding commands to compress the amount of data for each GLOBCNT pair. This section specifies details about how to encode and decode a **GLOBSET** during **IDSET** serialization.

Because compression is achieved by using a common byte stack to encode or decode the high order bytes of all GLOBCNT values, the encoder or decoder MUST construct a byte stack before implementing any of these commands.

3.1.5.5.3.1 Encoding

The commands specified in the following sections can be used to encode a **GLOBSET**.

Aside from the requirements set forth in this section, this specification does not mandate how the encoding and decoding commands are used. When more than one command can be used to achieve the same result set, the choice of command used is an implementation decision.

3.1.5.5.3.1.1 Push Command (0x01 – 0x06)

The **Push** command SHOULD be used when multiple GLOBCNT values share the same high-order values. For example, if all GLOBCNT values have the same two high-order bytes, use the **Push** command (0x02) to push two bytes onto the common byte stack. These two bytes will be used to create GLOBCNT pairs during decoding.

The **Push** command can also be used to generate an encoding for a singleton range where the low value and the high value are the same. When a **Push** command places a sixth byte onto the common byte stack, it tells the decoder the next GLOBCNT pair has all six bytes in common. This will place a singleton GLOBCNT range into the **GLOBSET** when decoded. The values added to the common byte stack on the last **Push** command are removed automatically and do not require a **Pop** command.

For more details about the format of the **Push** command, see section 2.2.2.5.1.

3.1.5.5.3.1.2 Pop Command (0x50)

Bytes that have been pushed onto the common byte stack with a **Push** command can be removed using the **Pop** command. The **Push** and **Pop** commands are used together to adjust the bytes that are stored on the common byte stack. The common byte stack is used to reduce the amount of serialized data if the GLOBCNT values all share common high-order bytes. This allows for those common high-order bytes to be encoded and placed into the serialization buffer only once and not repeated with every GLOBCNT. The **Pop** command MUST NOT be used if no bytes are currently on the common byte stack.

For more details about the format of the **Pop** command, see section 2.2.2.5.2.

3.1.5.5.3.1.3 Bitmask Command (0x42)

The **Bitmask** command is used when there are multiple GLOBCNT ranges that share five high-order bytes in common and the low-order bytes are all within 8 values of each other. Each GLOBCNT range is represented by one or more bits in a bitmask. There MUST already be five high-order bytes in the common byte stack to use this command. The **Bitmask** command can only represent at most five GLOBCNT ranges.

For more details about the format of the **Bitmask** command and its fields, see section 2.2.2.5.3.

The **StartingValue** field MUST be set to the low-order byte of the low value of the first GLOBCNT range. The **Bitmask** field MUST have one bit set for each value within a range, excluding the low value of the first GLOBCNT range. The bit to set for each value within a range is determined by subtracting the low-order byte of the GLOBCNT from one more than the **StartingValue**. This will produce a 0 based bit number value, where zero (0) is the lowest order bit and 7 is the highest

order bit in the **Bitmask** field. For all GLOBCNT values between ranges, the bit associated with the value is not set in the bitmask.

For example, given a set of ranges where all have the same five high-order bytes in common and the low-order bytes are the values {0x01-0x03, 0x05-0x05, 0x07-0x09}, it would be encoded as a **StartingValue** of 0x01 and the **Bitmask** would be 0xEB. The **Bitmask** value is broken down in the following table.

Low-Order Byte Value	0x09	0x08	0x07	0x06	0x05	0x04	0x03	0x02
Bit Number	7	6	5	4	3	2	1	0
Bit Value	1	1	1	0	1	0	1	1

If you take the **StartingValue** and each low-order byte value corresponding to a bit that is set in the **Bitmask**, you end up with the low-order byte values {0x01, 0x02, 0x03, 0x05, 0x07, 0x08, 0x09}. If you collapse these into ranges, you will have {0x01-0x03, 0x05-0x05, 0x07-0x09}.

3.1.5.5.3.1.4 Range Command (0x52)

The **Range** command is used to generate a single GLOBCNT range. If the low and high value of the GLOBCNT range are not the same, or the range has values that are more than 8 bytes from each other or the low and high value do not share five high-order bytes in common, the **Range** command MUST be used.

If the low and high GLOBCNT values share common high-order bytes, these SHOULD be pushed onto the common byte stack by using the **Push** command prior to using the **Range** command. The low-order bytes that are not in common are used to build the **Range** command.

For more details about the format of the **Range** command and its fields, see section 2.2.2.5.4.

3.1.5.5.3.1.5 End Command (0x00)

The **End** command is used to signal the end of the **GLOBSET** encoding. This command MUST be added after all GLOBCNT ranges within the **GLOBSET** have been encoded. The **End** command can only be used if the common byte stack is empty. If after all **GLOBCNT** ranges have been encoded, there are still bytes on the common byte stack, they MUST be removed with one or more **Pop** commands before the **End** command can be used.

For more details about the format of the **End** command, see section 2.2.2.5.5.

3.1.5.5.3.2 Decoding

The commands specified in this section can exist in a serialized **GLOBSET**. The server SHOULD send the client an RpcFormat error (0x000004B6), or MAY send a FormatError error (0x000004ED)<18>, as specified in [MS-OXCDATA] section 2.4.1, if it encounters an unsupported command or any other decoding failures.

3.1.5.5.3.2.1 Push Command (0x01 – 0x06)

The **Push** command can add one to six bytes of high-order bytes to a common byte stack. When a decoding role encounters a **Push** command during the decoding process, the decoder adds the number of bytes indicated by **Push** command to a common byte stack from highest to lowest byte order. The common byte stack is used in conjunction with subsequent encoding commands to build GLOBCNT pairs that represent GLOBCNT ranges within the **GLOBSET**. When building a GLOBCNT, all

the bytes on the common byte stack are used and any remaining bytes needed for a complete GLOBCNT have to come from the next encoding command in the stream.

For more details about the format of the **Push** command in the serialization buffer, see section 2.2.2.5.1.

3.1.5.5.3.2.2 Pop Command (0x50)

The **Pop** command removes the bytes that were previously pushed onto the common byte stack from the last **Push** command. The **Pop** command unwinds the stack in the reverse order in which the bytes where pushed.

For more details about the format of the **Pop** command in the serialization buffer, see section 2.2.2.5.2.

3.1.5.5.3.2.3 Bitmask Command (0x42)

The decoder only encounters the **Bitmask** command when there are five bytes in the common byte stack. The server SHOULD send the client an RpcFormat error (0x000004B6), and MAY send the client a FormatError error (0x000004ED)<19>, as specified in [MS-OXCDATA] section 2.4.1, if the decoder encounters the **Bitmask** command when there are more or fewer than five bytes in the common byte stack.

For more details about the format of the **Bitmask** command and its fields, see section 2.2.2.5.3.

Using the **StartingValue** and the **Bitmask** fields of the **Bitmask** command, a set of low-order bytes can be produced. For more details about decoding the **Bitmask** field to produce individual low-order values, see section 3.1.5.5.3.1.3. Each low-order byte MUST be combined with the required five high-order bytes on the common byte stack to form a complete 6-byte GLOBCNT value, which MUST be added to the **GLOBSET**.

3.1.5.5.3.2.4 Range Command (0x52)

The **Range** command generates a GLOBCNT range. The GLOBCNT range MUST be added to the **GLOBSET**.

For details about the format of the **Range** command and its fields, see section 2.2.2.5.4.

The **Range** command contains two byte array fields, the **LowValue** and **HighValue**. Each of these fields MUST be combined with any high-order bytes in the common byte stack to produce a 6-byte GLOBCNT value. The two GLOBCNT values are the low and high value of the GLOBCNT range.

The server SHOULD send the client an RpcFormat error (0x000004B6), and MAY send the client a FormatError error (0x000004ED)<20>, as specified in [MS-OXCDATA] section 2.4.1, if the high value of the range is larger than the low value of the range.

3.1.5.5.3.2.5 End Command (0x00)

When the **End** command is encountered, the **GLOBSET** MUST be complete based on the GLOBCNT values generated from any previous encoding commands.

3.1.5.6 Creating Compact IDSETs

As the number of changes that happen to a folder grows over its lifetime, the sets of **MID** values ([MS-OXCDATA] section 2.2.1.2) and **CN** values, as specified in section 2.2.2.1, that need to be kept in **IDSET**, as specified in section 2.2.2.4. grow as well. The size of the **IDSET** structure is

rarely a problem for hierarchy synchronization operations due to the small number of folders commonly present in mailboxes. Therefore, this discussion focuses on content synchronization operations. In this section, the term **IDSET** is used to refer to both **IDSETs** and **CNSETs**.

The following mechanisms are available to help optimize **IDSETs** for performance:

1. **IDSET** compression: The wire format of **IDSETs** is optimized for consecutive ranges and sets of nonconsecutive IDs that have close values.

2. Clustering of IDs: Clients and servers SHOULD allocate IDs of messages within a folder from contiguous sets of IDs. This optimization is based on an assumption that with time, all old messages will be either deleted or moved to another folder, and so all of their IDs could be represented as one range. For more details, see section 3.3.5.1.1.

3. Collapsing of ranges: If an **IDSET** is never iterated over and is only used in operations like "not in", it is possible to add ranges of IDs to the **IDSET** to help collapse its regions, if that would not affect the results of operations it is used in.

Note that because the synchronization scope limits synchronization to one folder, and the algorithm for determining the difference between replicas, as specified in section 3.2.5.1, only checks that a certain ID is not in the PidTagCnset * properties, it is possible to add **CN** values that were either never used or used on objects outside the synchronization scope to these **IDSETs** without affecting the outcome. Note that this MUST NOT be done for **IDSETs** that are ever iterated over, such as the **PidTagIdsetGiven** property (section 2.2.1.1.1), as it will change the outcome.

For example, an **IDSET** contains [10; 20] and [30; 40] for some REPLGUID. Because every internal change number within the same REPLGUID MUST be greater than any previous one, and the change numbers [21; 29] do not belong to any messages in the current folder, the two regions can be safely collapsed into [10; 40].

3.1.5.7 Calculating and Using PidTagMessageSize

A server SHOULD make the best effort to calculate this property, but because values for properties may change before the client downloads the message, and because the client specifies what data it does and does not require, it MUST be treated only as an estimate by client.

3.1.5.8 Using FastTransfer Streams in ROPs

The following table describes how possible root elements in the FastTransfer stream correspond to Bulk Data Transfer operations defined in section 2.2.3. Every download operation has to be configured prior to being able to produce a FastTransfer stream. Configuration starts by sending one of the ROPs in the following table and then performing the additional ROP specific configuration steps, as specified in sections 3.3.5.4.1 and 3.3.5.5.1.

ROP that initiates an operation	Root element in the produced FastTransfer stream	ROP request buffer field conditions
RopSynchronizationConfigure	contentsSync	**SynchronizationType** equals **Contents**.
	hierarchySync	**SynchronizationType** equals **Hierarchy**.
RopSynchronizationGetTransferState	state	Always.

ROP that initiates an operation	Root element in the produced FastTransfer stream	ROP request buffer field conditions
RopFastTranserSourceCopyTo **RopFastTranserSourceCopyProperties**	folderContent	**InputServerObject** is a Folder object.<21>
	messageContent	**InputServerObject** is a Message object.
	attachmentContent	**InputServerObject** is an Attachment object.<22>
RopFastTranserSourceCopyMessages	messageList	Always.
RopFastTranserSourceCopyFolder	topFolder	Always.

FastTransfer streams produced by operations initiated by the **RopSynchronizationConfigure** ROP are intended for processing on the client only.

FastTransfer streams produced by operations initiated with the **RopFastTransferSource*** ROPs can either be processed by the client or uploaded to the server through an operation initiated by the **RopFastTransferDestinationConfigure** ROP. For details about the applicability of FastTransfer streams to FastTransfer upload operations, see section 2.2.3.1.2.1.1.

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 following abstract object types are defined in this document:

Global

3.2.1.1 Global

There are no global parameters defined as specific to the server.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Determining What Differences Need to be Downloaded

In this section, all references to the ICS state properties refer to values uploaded in the initial ICS state.

For every object in the synchronization scope, servers MUST do the following:

- Include information about a change to an object if one of the following applies:
  - It is a folder and a change number is not in **PidTagCnsetSeen** property (section 2.2.1.1.2).
  - It is a normal message
    - and **SynchronizationFlag Normal** was set
    - and a change number is not in the **PidTagCnsetSeen** property.
  - It is an FAI message, meaning the **PidTagAssociated** property (section 2.2.1.4) is set to "TRUE")
    - and **SynchronizationFlag FAI** was set
    - and a change number is not in the **PidTagCnsetSeenFAI** property (section 2.2.1.1.3).
- If **SynchronizationFlag NoDeletions** is not set, include deletion information about objects that either:
  - Have their internal identifiers (2) present in the **PidTagIdsetGiven** property (section 2.2.1.1.1)
    - and are missing from the server replica.
  - Are folders that have never been reported as deleted.
- If **SynchronizationFlag NoDeletion** and **IgnoreNoLongerInScope** are not set, include deletion information about messages that went out of scope that:
  - Have their internal identifiers (2) present in **PidTagIdsetGiven**
    - and exist in a server replica and belong to a folder that defines the synchronization scope
    - and do not match the restriction that defines the synchronization scope.
- If **SynchronizationFlag ReadState** is set, include read state change information about messages that:
  - Do not have their change numbers for read and unread state in the **PidTagCnsetRead** property (section 2.2.1.1.4)

- and are not FAI messagesand have not had change information downloaded for them in this session.

<23>

The following invariants define the relationship between the initial ICS state, the checkpoint ICS state, and differences downloaded at the time of checkpointing. The following table contains the nomenclature used to describe the invariants. For more details about checkpointing, see section 2.2.3 and 3.3.5.3.

Nomenclature	Description
PropIndex	Property **Prop** of the ICS state, as specified in section 2.2.1.1. **Index** can be I for initial and C for checkpoint.
PropD	Property **Prop** that contains a particular set of differences that have been downloaded in the current operation, as specified in section 2.2.1.3.
{change Subset**.Id}** **{change**Subset**.CN}**	Internal identifiers (2) (Id) or change numbers of all changes that have been downloaded in the current operation. The **Subset** can be one of the following: ▪ Omitted to denote all changes. ▪ **Normal** for normal messages. ▪ FAI for FAI messages. ▪ **Partial** for normal messages downloaded as partial changes.
{readStateChange.Id} **{readStateChange.ReadStateCn}**	Internal identifiers (2) or read state change numbers of all normal messages, with only the read state changed, which have been downloaded in the current operation.

Servers MUST ensure that the following invariants are true:

- AllDeleted = (IdsetDeletedD ∪ IdsetNoLongerInScopeD ∪ IdsetExpiredD)
- IdsetGivenC = (IdsetGivenI ∪ {change.Id}) \ AllDeleted
- CnsetSeenC = CnsetSeenI ∪ {changeNormal.Cn}
- CnsetSeenFAIC = CnsetSeenFAII ∪ {changeFAI.Cn}
- CnsetReadC = CnsetReadI ∪ {readStateChange.ReadCn}
- IdsetGivenI ⊇ {changesPartial.Id}
- IdsetGivenI ⊇ (IdsetReadD ∪ IdsetUnreadD)
- {readStateChange.Id} = IdsetReadD ∪ IdsetUnreadD
- {change.Id} ∩ AllDeleted = ø
- {change.Cn} ∩ (CnsetSeenI ∪ CnsetSeenFAII) = ø
- {readStateChange.Id} ∩ AllDeleted = ø

- {readStateChange.Id} ∩ {change.Id} = ø

3.2.5.2 Generating the PidTagSourceKey Value

When requested by the client, the server MUST output the **PidTagSourceKey** property (section 2.2.1.2.5) value if it is persisted, or generate it on-the-fly if it is missing. If the **PidTagSourceKey** value is missing, the server MUST generate it by producing a **GID** value ([MS-OXCDATA] section 2.2.1.3) from the internal identifier (2) (**MID** ([MS-OXCDATA] section 2.2.1.2) or **FID** ([MS-OXCDATA] section 2.2.1.1)) of the object by using the same mapping algorithm as described for the **RopLongTermIdFromId** ROP, as specified in [MS-OXCSTOR].

The only exception is when a server is required to generate this property on the fly for a folder, which is a root of the current hierarchy synchronization download operation (that is, it is the folder that was passed to the **RopSynchronizationConfigure** ROP). In this case, **PidTagSourceKey** MUST be output as a zero-length **PtypBinary**, as specified in [MS-OXCDATA] section 2.11.1.

3.2.5.3 Tracking Read State Changes

To conserve the bandwidth between clients and servers, the read state of the messages SHOULD be tracked separately from other changes.

Whenever the read state of a message changes on the server, a separate change number (the read state change number) on the message SHOULD be assigned a new value. The change number of the message SHOULD NOT be modified unless other changes to a message were made at the same time. This allows the change to be efficiently downloaded to a client as the **MID** value ([MS-OXCDATA] section 2.2.1.2) in the **PidTagIdsetRead** property (section 2.2.1.3.4) **IDSET** or the **PidTagIdsetUnread** property (section 2.2.1.3.5), compressed together with read state changes to other messages in the synchronization scope. An individual read state change number is never sent across the wire independently. An **IDSET** of change numbers associated with message read state transitions, either from read to unread, or unread to read (as determined by the **PidTagMessageFlags** property in [MS-OXCMSG] section 2.2.1.6) are included in the **PidTagCnsetRead** property (section 2.2.1.1.4), which is part of the ICS state and is never directly set on any objects.

3.2.5.4 Receiving FastTransfer ROPs

3.2.5.4.1 Download

When producing FastTransfer streams for operations configured with **RopFastTransferSourceCopy*** ROPs, servers SHOULD skip over objects that the client does not have adequate permissions for. For example, if the **Move** flag of the **CopyFlags** field, as specified in section 2.2.3.1.1.1.1, is set, an additional permission to delete an object is required for the object to be included in the output FastTransfer stream. If a permission check for an object fails, the **PidTagEcWarning** meta-property (section 2.2.4.1.5.2) SHOULD be output in a FastTransfer stream, wherever allowed by its syntactical structure, to signal a client about incomplete content.

3.2.5.4.1.1 Receiving a RopFastTransferSourceCopyTo Request

When the client sends the server a **RopFastTransferSourceCopyTo** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.6.1 and section 2.2.3.1.1.1 of this specification. The server MUST respond with a **RopFastTransferSourceCopyTo** ROP response, as specified in [MS-OXCROPS] section 2.2.13.6.2 and section 2.2.3.1.1.1 of this specification.

If the **CopyFlags Move** flag is set, the server SHOULD NOT output any objects in a FastTransfer stream that the client does not have permissions to delete.

If the **CopyFlags BestBody** flag is set, the server SHOULD output the message body (2), and the body of the Embedded Message object, in their original format. If this flag is not set, the server MUST output the message body (2) in the compressed Rich Text Format (RTF).

Servers SHOULD fail the ROP if unknown **CopyFlags** flag bits are set.

If the server supports partial message downloads and the **SendOptions PartialItem** flag is set, the server SHOULD output partial message changes if it reduces the size of the produced stream.

If the server does not support partial message downloads, it does not output partial message changes and the **SendOptions PartialItem** flag is ignored.

Servers SHOULD fail the ROP if any unknown **SendOptions** flag bits are set.

3.2.5.4.1.2 Receiving a RopFastTransferSourceCopyProperties Request

When the client sends the server a **RopFastTransferSourceCopyProperties** ROP request, the server MUST parse the reques,t as specified in [MS-OXCROPS] section 2.2.12.7.1 and section 2.2.3.1.1.2 of this specification. The server MUST respond with a **RopFastTransferSourceCopyProperties** ROP response, as specified in [MS-OXCROPS] section 2.2.12.7.2 and section 2.2.3.1.1.2 of this specification.

If the **CopyFlags Move** flag is specified for a download operation, the server SHOULD NOT output any objects in a FastTransfer stream that the client does not have permissions to delete.

Servers SHOULD fail the ROP if unknown **CopyFlag** flag bits are set.

If the server supports partial message downloads and the **SendOptions PartialItem** flag is set, the server SHOULD output partial message changes if it reduces the size of the produced stream.

If the server does not support partial message downloads, it does not output partial message changes and the **SendOptions PartialItem** flag is ignored.

Servers SHOULD fail the ROP if any unknown **SendOptions** flag bits are set.

3.2.5.4.1.3 Receiving a RopFastTransferSourceCopyMessage Request

When the client sends the server a **RopFastTransferSourceCopyMessage** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.12.5.1 and section 2.2.3.1.1.3 of this specification. The server MUST respond with a **RopFastTransferSourceCopyMessage** ROP response, as specified in [MS-OXCROPS] section 2.2.12.5.2 and section 2.2.3.1.1.3 of this specification.

If the **CopyFlags Move** flag is set for a download operation, the server SHOULD NOT output any objects in a FastTransfer stream that the client does not have permissions to delete.

If the **CopyFlags BestBody** flag is set, the server SHOULD output the message body (2), and the body of the Embedded Message object, in their original format.

If the **CopyFlags BestBody** flag not set, the server MUST output message bodies (2) in the compressed RTF.

If the server supports partial message downloads and the **SendOptions PartialItem** flag is set, the server SHOULD output partial message changes if it reduces the size of the produced stream.

If the server does not support partial message downloads, it does not output partial message changes and the **SendOptions PartialItem** flag is ignored.

Servers SHOULD fail the ROP if any unknown **SendOptions** flag bits are set.

3.2.5.4.1.4 Receiving a RopFastTransferSourceCopyFolder Request

When the client sends the server a **RopFastTransferSourceCopyFolder** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.12.4.1 and section 2.2.3.1.1.4 of this specification. The server MUST respond with a **RopFastTransferSourceCopyFolder** ROP response, as specified in [MS-OXCROPS] section 2.2.12.4.2 and section 2.2.3.1.1.4 of this specification.

If the **CopyFlags CopySubfolders** flag is set, the server MUST recursively include the subfolders of the folder specified in the **InputServerObject** in the scope.

If the **CopyFlags CopySubfolders** flag is not set, the server MUST NOT recursively include the subfolders of the folder specified in the **InputServerObject** in the scope.

If the **CopyFlags NoGhostedContent** flag is set and the folder is ghosted, the server SHOULD send the folder properties, but SHOULD NOT send the content of the ghosted folder.

If the **CopyFlags NoGhostedContent** flag is not set and the folder is ghosted, the server SHOULD return the folder properties, as specified in section 2.2.4.1.5.3. The **PidTagNewFXFolder** property (section 2.2.4.1.5.3) contains information about the location of servers that contain replica content.

If the **CopyFlags NoGhostedContent** flag is set on a nonghosted folder, the server SHOULD send the folder properties and the folder content.

Servers SHOULD fail the ROP if unknown **CopyFlags** flag bits are set.

If the server supports partial message downloads and the **SendOptions PartialItem** flag is set, the server SHOULD output partial message changes if it reduces the size of the produced stream.

If the server does not support partial message downloads, it does not output partial message changes and the **SendOptions PartialItem** flag is ignored.

Servers SHOULD fail the ROP if any unknown **SendOptions** flag bits are set.

3.2.5.4.1.5 Receiving a RopFastTransferSourceGetBuffer Request

When the client sends the server a **RopFastTransferSourceGetBuffer** ROP request, the server MUST parse the request as specified in [MS-OXCROPS] section 2.2.12.6.1 and section 2.2.3.1.1.1 of this specification. The server MUST respond with a **RopFastTransferSourceGetBuffer** ROP response, as specified in [MS-OXCROPS] section 2.2.12.6.2 and section 2.2.3.1.1.1 of this specification.

If the value of the **BufferSize** in the ROP request is 0xBABE, the server determines the buffer size based on the residual size of the RPC buffer.

If the value of the **BufferSize** in the ROP request is set to 0xBABE, the server MUST limit the amount of data returned in **TransferBuffer** to the residual size of the output buffer minus result structure overhead, or limit the amount of data returned in **TransferBuffer** to **MaximumBufferSize**, whichever is smaller.

If the value of **BufferSize** in the ROP request is set to a value other than 0xBABE, the following semantics apply:

- The server MUST fail the command before processing the ROP by doing the following:

- Failing the entire RPC with **ReturnValue ecBufferTooSmall** (0x0000047D) if the server is not able to fit the resulting **BufferSize** bytes in **TransferBuffer** into the biggest possible output RPC buffer allowed by the protocol based on the response buffer size requested by the client. The absolute maximum size of the output buffer is 32743 bytes.
- Returning the **RopBufferTooSmall** ROP if the server will not be able to fit the resulting **BufferSize** bytes in **TransferBuffer** into the residual output RPC buffer.

- The server MUST output, at most, **BufferSize** bytes in the **TransferBuffer** even if more data is available.
- The server returns less than or equal to the **BufferSize** bytes in **TransferBuffer**.

A **ReturnValue** of **ServerBusy** should only be returned when the client is version 11.0.0.4920 or higher. For more details about version checking, see [MS-OXCRPC] section 3.1.9.3.

Servers SHOULD fail any successive calls to the **RopFastTransferSourceGetBuffer** ROP, after the previous iteration returns a buffer with a **ReturnValue** other that **Success** or **ServerBusy**.

3.2.5.4.1.6 Receiving a RopTellVersion Request

When the client sends the server a **RopTellVersion** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.12.8.1 and section 2.2.3.1.1.6 of this specification. The server MUST respond with a **RopTellVersion** ROP response, as specified in [MS-OXCROPS] section 2.2.12.8.2 and section 2.2.3.1.1.6 of this specification.

3.2.5.4.2 Upload

3.2.5.4.2.1 Receiving a RopFastTransferDestinationConfigure Request

When the client sends the server a **RopFastTransferDestinationConfigure** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.12.1.1 and section 2.2.3.1.2.1 of this specification. The server MUST respond with a **RopFastTransferDestinationConfigure** ROP response, as specified in [MS-OXCROPS] section 2.2.12.1.2 and section 2.2.3.1.2.1 of this specification.

Any changes to an object identified by **InputServerObject** in the ROP request are not persisted until the **RopSaveChangesMessage** ROP is called.

The server MUST stop execution of the ROP if an unknown **SourceOperation** value is passed.

The server SHOULD fail the ROP if unknown **CopyFlags** bits are set.

3.2.5.4.2.2 Receiving a RopFastTransferDestinationPutBuffer Request

When the client sends the server a **RopFastTransferDestinationPutBuffer** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.12.2.1 and section 2.2.3.1.2.2 of this specification. The server MUST respond with a **RopFastTransferDestinationPutBuffer** ROP response, as specified in [MS-OXCROPS] section 2.2.12.2.2 and section 2.2.3.1.2.2 of this specification.

3.2.5.5 Receiving Incremental Change Synchronization ROPs

3.2.5.5.1 Download

3.2.5.5.1.1 Receiving a RopSynchronizationConfigure Request

When the client sends the server a **RopSynchronizationConfigure** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.1.1 and section 2.2.3.2.1.1 of this specification. The server MUST respond with a **RopSynchronizationConfigure** ROP response, as specified in [MS-OXCROPS] section 2.2.13.1.2 and section 2.2.3.2.1.1 of this specification.

SynchronizationType Constraints

The following constraints apply to the **SynchronizationType** field.

Servers MUST fail the ROP if an unknown **SynchronizationType** value is passed.

If the **SynchronizationFlag Unicode** flag is set, the client supports Unicode and the server MUST output values of string properties as they are stored, whether in Unicode or non-Unicode format.

If the **SynchronizationFlag Unicode** flag is not set, the client does not support Unicode and the server MUST output values of string properties in the code page set on connection.

If the **SynchronizationFlag NoDeletions** flag is set, the server MUST NOT download information about item deletions, as specified in section 2.2.4.3.3, and the server MUST behave as if **IgnoreNoLongerInScope** was set.

If the **SynchronizationFlag NoDeletions** flag is not set, the server MUST download information about item deletions, as specified in section 2.2.4.3.3.

If the **SynchronizationFlag IgnoreNoLongerInScope** flag is set, the server MUST NOT download information about messages that went out of scope as deletions, as specified in section 2.2.4.3.3.

If the **SynchronizationFlag IgnoreNoLongerInScope** flag is not set, the server MUST download information about messages that went out of scope as deletions, as specified in section 2.2.4.3.3.

If the **SynchronizationFlag ReadState** flag is set, the server MUST also download information about changes to the read state of messages, as specified in section 2.2.4.3.22.

If the **SynchronizationFlag ReadState** flag is not set, the server MUST NOT download information about changes to the read state of messages, as specified in section 2.2.4.3.22.

If the **SynchronizationFlag FAI** flag is set, the server MUST download information about changes to FAI messages, as specified in section 2.2.4.3.7.

If the **SynchronizationFlag FAI** flag is not set, the server MUST NOT download information about changes to FAI messages, as specified in section 2.2.4.3.7.

If the **SynchronizationFlag Normal** flag is set, the server MUST download information about changes to normal messages, as specified in section 2.2.4.3.11.

If the **SynchronizationFlag Normal** flag is not set, the server MUST NOT download information about changes to normal messages, as specified in section 2.2.4.3.11.

If the **SynchronizationFlag OnlySpecifiedProperties** flag is set, the server SHOULD limit properties and subobjects output for **top-level messages** to the properties listed in **PropertyTags**.

If the **SynchronizationFlag OnlySpecifiedProperties** flag is not set, the server SHOULD exclude properties and subobjects output for folders and top-level messages, if they are listed in **PropertyTags**.

If the **SynchronizationFlag NoForeignIdentifiers** flag is set, the server MUST ignore any persisted values for the **PidTagSourceKey** property (section 2.2.1.2.5) and **PidTagParentSourceKey** (section 2.2.1.2.6) properties when producing output for folder and message changes.

If the **SynchronizationFlag NoForeignIdentifiers** flag is not set, the server MUST NOT ignore any persisted values for the **PidTagSourceKey** and **PidTagParentSourceKey** properties when producing output for folder and message changes.

The server MUST fail the ROP request if the **SynchronizationFlag Reserved** flag is set.

If the **SynchronizationFlag BestBody** flag is set, the server SHOULD<24> output message bodies (2) in their original format.

If the **SynchronizationFlag BestBody** flag is not set, the server MUST output message bodies (2) in the compressed RTF format.

If the **SynchronizationFlag IgnoreSpecifiedOnFAI** flag is set, the server MUST output all properties and subobjects of FAI messages.

If the **SynchronizationFlag IgnoreSpecifiedOnFAI** flag is not set, the server ignores properties and subobjects of FAI messages.

If the **SynchronizationFlag Progress** flag is set, the server SHOULD inject the **progressTotal** element, as specified in section 2.2.4.3.19, into the output FastTransfer stream.

If the **SynchronizationFlag Progress** flag is not set, the server MUST not inject the **progressTotal** element into the output FastTransfer stream.

Servers SHOULD<25> fail the ROP if unknown flag bits are set.

SynchronizationExtraFlag Constraints

The following constraints apply to the **SynchronizationExtraFlag** field.

The server MUST include the **PidTagFolderId** property (section 2.2.1.2.2) (for hierarchy synchronization operations) or the **PidTagMid** property (section 2.2.1.2.1) (for content synchronization operations) in a folder change or message change header if and only if the **SynchronizationExtraFlag Eid** flag is set.

The server MUST NOT include the **PidTagFolderId** property (for hierarchy synchronization operations) or the **PidTagMid** property (for content synchronization operations) in a folder change or message change header if and only if the **SynchronizationExtraFlag Eid** flag is not set.

The server MUST include the **PidTagMessageSize** property (section 2.2.1.5) in the message change header if and only if the **SynchronizationExtraFlag MessageSize** flag is set.

The server MUST include the **PidTagChangeNumber** property (section 2.2.1.2.3) in the message change header if and only if the **SynchronizationExtraFlag CN** flag is set.

The server MUST NOT include the **PidTagChangeNumber** property in the message change header if and only if the **SynchronizationExtraFlag CN** flag is not set.

If the **SynchronizationExtraFlag OrderByDeliveryTime** flag is set, the server MUST sort messages by the value of their **PidTagMessageDeliveryTime** property ([MS-OXOMSG] section 2.2.3.9), or by the **PidTagLastModificationTime** property ([MS-OXPROPS] section 2.861) if the former is missing, when generating a sequence of **messageChange** elements for the FastTransfer stream, as specified in section 2.2.4.2.

If the **SynchronizationExtraFlag OrderByDeliveryTime** flag is not set, there is no requirement on the server to return items in a specific order.

Servers MUST ignore any unknown **SynchronizationExtraFlag** flag bits.

PropertyTags Constraints

The following constraints apply to the **PropertyTags** field.

This field has different semantics, depending on the value of the **SynchronizationFlag OnlySpecifiedProperties** flag, as follows:

- If the **OnlySpecifiedProperties** flag is not set, the server SHOULD exclude properties and subobjects from output for folders and top-level messages, if the property is listed in the **PropertyTags** field.
- If the **OnlySpecifiedProperties** flag is set, the server SHOULD limit properties and subobjects output for top-level messages to properties listed in the **PropertyTags** field.

Inclusion of properties that denote message subobjects in the **PropertyTags** field means that the server SHOULD include or exclude these special parts from output for top-level messages.

3.2.5.5.2 Uploading State

3.2.5.5.2.1 Receiving a RopSynchronizationUploadStateStreamBegin Request

When the client sends the server a **RopSynchronizationUploadStateStreamBegin** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.9.1 and section 2.2.3.2.2.1 of this specification. The server MUST respond with a **RopSynchronizationUploadStateStreamBegin** ROP response, as specified in [MS-OXCROPS] section 2.2.13.9.2 and section 2.2.3.2.2.1 of this specification.

3.2.5.5.2.2 Receiving a RopSynchronizationUploadStateStreamContinue Request

When the client sends the server a **RopSynchronizationUploadStateStreamContinue** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.10.1 and section 2.2.3.2.2.2 of this specification. The server MUST respond with a **RopSynchronizationUploadStateStreamContinue** ROP response, as specified in [MS-OXCROPS] section 2.2.13.10.2 and section 2.2.3.2.2.2 of this specification.

Servers concatenate **StreamData** from all received **RopSynchronizationUploadStateStreamContinue** ROP requests for a given ICS state property.

3.2.5.5.2.3 Receiving a RopSynchronizationUploadStateStreamEnd Request

When the client sends the server a **RopSynchronizationUploadStateStreamEnd** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.11.1 and section 2.2.3.2.2.3 of this specification. The server MUST respond with a **RopSynchronizationUploadStateStreamEnd** ROP response, as specified in [MS-OXCROPS] section 2.2.13.11.2 and section 2.2.3.2.2.3 of this specification.

3.2.5.5.3 Downloading State

3.2.5.5.3.1 Receiving a RopSynchronizationGetTransferState Request

When the client sends the server a **RopSynchronizationGetTransferState** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.8.1 and section 2.2.3.2.3.1 of this specification. The server MUST respond with a **RopSynchronizationGetTransferState** ROP response, as specified in [MS-OXCROPS] section 2.2.13.8.2 and section 2.2.3.2.3.1 of this specification.

The server MUST ensure that changes to the state of the synchronization context that occur after this ROP do not affect the ICS state that is downloaded through the FastTransfer download context that is returned from this ROP.

3.2.5.5.4 Upload

3.2.5.5.4.1 Receiving a RopSynchronizationOpenCollector Request

When the client sends the server a **RopSynchronizationOpenCollector** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.7.1 and section 2.2.3.2.4.1 of this specification. The server MUST respond with a **RopSynchronizationOpenCollector** ROP response, as specified in [MS-OXCROPS] section 2.2.13.7.2 and section 2.2.3.2.4.1 of this specification.

3.2.5.5.4.2 Receiving a RopSynchronizationImportMessageChange Request

When the client sends the server a **RopSynchronizationImportMessageChange** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.2.1 and section 2.2.3.2.4.2 of this specification. The server MUST respond with a **RopSynchronizationImportMessageChange** ROP response, as specified in [MS-OXCROPS] section 2.2.13.2.2 or 2.2.13.2.3, and in section 2.2.3.2.4.2 of this specification.

The server is responsible for conflict detection and resolution, as specified in section 3.1.4.1.

The server MUST detect conflicts. Conflict resolution is controlled by the value of the **PidTagResolveMethod** property (section 2.2.1.6) set on the containing folder. If a conflict was detected, a ROP can succeed and return a handle to a Message object in the response buffer. The server becomes responsible for performing conflict resolution on the **RopSaveChangesMessage** ROP, as specified in section 3.1.4.1.2.

Servers SHOULD fail the ROP if unknown flag bits are set.

Upon successful completion of the **RopSynchronizationImportMessageChange** ROP, the ICS state on the synchronization context MUST be updated to include a new change number in either the **PidTagCnsetSeen** (section 2.2.1.1.2) or **PidTagCnsetSeenFAI** (section 2.2.1.1.3) property, depending on whether the particular message is a normal message or an FAI message.

The server MUST purge all client-settable properties and subobjects of the Message object prior to returning it in the **OutputServerObject**. Note that any changes to this message made by this ROP or any other ROP that operates on it MUST NOT be persisted until **RopSaveChangesMessage** ROP is called.

ImportFlag Constraints

If the **ImportFlag FailOnConflict** flag is set, the server MUST NOT accept conflicting versions of messages.

If the **ImportFlag FailOnConflict** flag is not set, the server MUST accept conflicting versions of messages.

3.2.5.5.4.3 Receiving a RopSynchronizationImportHierarchyChange Request

When the client sends the server a **RopSynchronizationImportHierarchyChange** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.4.1 and section 2.2.3.2.4.3 of this specification. The server MUST respond with a **RopSynchronizationImportHierarchyChange** ROP response, as specified in [MS-OXCROPS] section 2.2.13.4.2 or 2.2.13.4.3, and section 2.2.3.2.4.3 of this specification.

Upon successful completion of this ROP, the ICS state on the synchronization context MUST be updated to include a new change number in the **PidTagCnsetSeen** property (section 2.2.1.1.2).

The server is responsible for conflict detection and resolution, as specified in section 3.1.4.1.

If a conflict is detected, the server MUST resolve it as specified in section 3.1.4.1.2 and return Success. A server can report a conflict using a conflict notification message.

If a conflict has occurred, the server:

- SHOULD NOT update the **PidTagCnsetSeen** property, and let the clients download a result of conflict resolution.
- MAY generate a conflict notification message. For more details, see section 3.1.4.1.3.
- MUST return **Success** in the **ReturnValue**.

The server MUST ignore the properties in **PropertyValues**, which are also present in **HierarchyValues**.

3.2.5.5.4.4 Receiving a RopSynchronizationImportMessageMove Request

When the client sends the server a **RopSynchronizationImportMessageMove** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.6.1 and section 2.2.3.2.4.4 of this specification. The server MUST respond with a **RopSynchronizationImportMessageMove** ROP response, as specified in [MS-OXCROPS] section 2.2.13.6.2 or 2.2.13.6.3, and section 2.2.3.2.4.4 of this specification.

Upon successful completion of this ROP, the ICS state on the synchronization context MUST be updated to include change numbers of messages in the destination folder in either the **PidTagCnsetSeen** (section 2.2.1.1.2) or **PidTagCnsetSeenFAI** (section 2.2.1.1.3) property, depending on whether the message is a normal message or an FAI message.

3.2.5.5.4.5 Receiving a RopSynchronizationImportDeletes Request

When the client sends the server a **RopSynchronizationImportDeletes** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.5.1 and section 2.2.3.2.4.5 of this specification. The server MUST respond with a **RopSynchronizationImportDeletes** ROP response, as specified in [MS-OXCROPS] section 2.2.13.5.2 and section 2.2.3.2.4.5 of this specification.

The server MUST ignore requests to delete objects that have already been deleted and SHOULD record deletions of objects that never existed in the server replica, in order to prevent the **RopSynchronizationImportHierarchyChange** or **RopSynchronizationImportMessageChange** ROPs from restoring them back.

The protocol does not dictate that deletions of all objects passed in the request to this ROP MUST happen in a transacted way. However, to minimize the possibility of putting replicas into a desynchronized state and because the protocol does not let clients know in any way what part of an operation has succeeded, servers bear the responsibility of making a reasonable prediction as to whether all deletions will succeed, and if a deletion will not succeed, report a failure right away, instead of partially completing an operation.

Servers SHOULD fail the ROP if unknown **ImportDeleteFlags** flag bits are set.

3.2.5.5.4.6 Receiving a RopSynchronizationImportReadStateChanges Request

When the client sends the server a **RopSynchronizationImportReadStateChanges** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.3.1 and section 2.2.3.2.4.6 of this specification. The server MUST respond with a **RopSynchronizationImportReadStateChanges** response, as specified in [MS-OXCROPS] section 2.2.13.3.2 and section 2.2.3.2.4.6 of this specification.

The **RopSynchronizationImportReadStateChanges** ROP is a batch variant of the **RopSetMessageReadFlag** ROP, which also takes care of updating the ICS state. The net effect of changing the read state message by message by using the **RopSetMessageReadFlag** ROP MUST be identical to changing the read state in bulk by using the **RopSynchronizationImportReadStateChanges** ROP.

Requests to change the read state of FAI messages MUST be ignored. Upon successful completion of this ROP, the ICS state on the synchronization context MUST be updated by adding the new change number to the **PidTagCnsetRead** property (section 2.2.1.1.4).

The protocol does not dictate that the change of the read state for all objects passed in the ROP request MUST happen in a transacted way. However, to minimize the possibility of putting replicas into a desynchronized state and because the protocol does not let clients know what part of an operation has succeeded, servers bear the responsibility of making a reasonable prediction as to whether changes of read state for all normal messages will succeed, and if the changes of read state will not succeed, report a failure immediately, instead of partially completing an operation.

3.2.5.5.4.7 Receiving a RopGetLocalReplicaIds Request

When the client sends the server a **RopGetLocalReplicaIds** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.13.1 and section 2.2.3.2.4.7 of this specification. The server MUST respond with a **RopGetLocalReplicaIds** ROP response, as specified in [MS-OXCROPS] section 2.2.13.13.2 or 2.2.13.13.3, and section 2.2.3.2.4.7 of this specification.

A server can limit the number of IDs that can be allocated in one batch to prevent malicious clients from reserving too many IDs with the intent of causing a denial-of-service attack by depleting the set of available IDs. A server can limit the maximum number of IDs that can be allocated in one batch to the upper limit of the range recommended to clients, as specified in section 3.3.5.5.4.9.

3.2.5.5.4.8 Receiving a RopSetLocalReplicaMidsetDeleted Request

When the client sends the server a **RopSetLocalReplicaMidsetDeleted** ROP request, the server MUST parse the request, as specified in [MS-OXCROPS] section 2.2.13.12.1 and section 2.2.3.2.4.8 of this specification. The server MUST respond with a **RopSetLocalReplicaMidsetDeleted** ROP response, as specified in [MS-OXCROPS] section 2.2.13.12.2 and section 2.2.3.2.4.8 of this specification.

A server MUST add ranges of IDs supplied through this ROP to the **deleted item list**. By adding ranges of IDs to the deleted item list, the server is able to compress the deleted item list by using the **IDSET** optimization algorithm specified in section 3.1.5.6.

A server MUST ensure that ranges supplied as request fields to this ROP are allocated by using the **RopGetLocalReplicaIds** ROP.

3.2.5.6 Effect of Property and Subobject Filters on Download

Property and subobject filters specified during the configuration of a download operation only have an effect on the objects that are directly included in the scope of the operation. For example:

- Specifying a property in the **PropertyTags** field of the request buffer of a **RopFastTransferSourceCopyProperties** ROP that is configured with an Attachment object as an **InputServerObject** will affect the set of properties to be copied for this attachment, but not its embedded message or any attachments that it might contain.
- Specifying the **PidTagFolderAssociatedContents** property ([MS-OXPROPS] section 2.775) in the **PropertyTags** field of the request buffer of a **RopFastTransferSourceCopyTo** ROP that is configured with a Folder object as an **InputServerObject** will only exclude FAI Message objects from copying this specific folder, but not any of its descendant folders.
- Specifying the **PidTagMessageRecipients** property ([MS-OXPROPS] section 2.892) in the **PropertyTags** fields of the request buffer of a **RopSynchronizationConfigure** ROP will exclude recipient subobjects from all message changes downloaded in that operation, but it will not affect recipients (1) of embedded messages that their attachments might have.

Regardless of property filters specified at operation configuration time, certain properties MUST always be excluded from output. For more details, see section 3.2.5.8.

At the same time, directives to include or exclude properties and subobjects supplied through flags do have an effect on downloaded objects at all levels. For example:

- Specifying the **CopyFlag CopySubfolders** flag, as specified in section 2.2.3.1.1.4.1, includes all subfolders of the current folder into the operation scope.
- Specifying **CopyFlag SendEntryId** flag includes all identification properties for all objects being downloaded.

Whenever subobject filters have an effect, servers MUST output a **PidTagFXDelProp** meta-property (section 2.2.4.1.5.1) immediately before outputting subobjects of a particular type, to differentiate between the cases where a set of subobjects (such as attachments or recipients (1)) was filtered in, but was empty, and where it was filtered out. For example:

- Specifying meta-property **PidTagMessageRecipients** property ([MS-OXPROPS] section 2.892) in the **PropertyTags** field of the request buffer of the **RopFastTransferSourceCopyProperties** ROP that is configured with a Message object as an **InputServerObject**, will direct the server to output the **PidTagFXDelProp** (section 2.2.4.1.5.1) and **PidTagMessageRecipients** properties before outputting recipients (1) of that message, even if there are no recipients (1).

The protocol does not support incremental download of subobjects. Subobjects of a particular type are either filtered out, in which case the **PidTagFXDelProp** meta-property MUST NOT be output, or are filtered in; that is, they MUST be output one after another, prefixed by the **PidTagFXDelProp** meta-property.

3.2.5.7 Properties to Ignore on Upload

Unless specified otherwise in property list restriction tables, properties that belong to the **provider-defined internal nontransmittable** range, as specified in [MS-OXPROPS] section 1.3.3, MUST be ignored on upload.

3.2.5.8 Properties to Ignore on Download

Unless specified otherwise in property list restriction tables, **propValue** elements of FastTransfer streams that belong to the provider-defined internal nontransmittable range, as specified in [MS-OXPROPS] section 1.3.3, MUST be excluded from download.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client Details

This section provides client-specific details related to bulk data transfer. The Mailbox Synchronization Protocol Specification, as specified in [MS-OXCSYNC], also contains important client-specific details related to bulk data transfer.

3.3.1 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 following abstract object types are defined in this document:

Global

Messaging Object

3.3.1.1 Global

There are no global parameters defined as specific to the client.

3.3.1.2 Per Messaging Object

Messaging objects are represented by the **MessagingObject** abstract object type. The following abstract object elements are maintained by the client for each **MessagingObject**:

Client.MessagingObject.ForeignIdentifier: An **XID**, as specified in section 2.2.2.2, that identifies changes to objects in the local replica.

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Creating Objects and Identifying Changes on the Local Replica

The following three alternative mechanisms are available to clients that require the ability to create objects in their local replica without having immediate contact with the server to upload the differences. This is also known as working **offline**.

3.3.5.1.1 Client-Assigned Internal Identifiers

When using this most preferred approach, clients MUST send a request to a server to allocate a range of internal identifiers (2) for their exclusive use by using **RopGetLocalReplicaIds** ROP. Once the range is allocated, a client can stay offline and use identifiers from that range until the range is exhausted, at which point the client would have to allocate a new range by connecting to the server and executing the **RopGetLocalReplicaIds** ROP before being able to assign new client-assigned internal identifiers (2). Clients can then assign these IDs to any new folders or messages within their local replica and communicate these assignments back when performing ICS upload by using the **RopSynchronizationImportHierarchyChange** ROP (section 2.2.3.2.4.3) or the **RopSynchronizationImportMessageChange** ROP (section 2.2.3.2.4.2). Note that these IDs MUST NOT be used for change numbers as it would result in change numbers that did not increase per namespace replica, or clients having different ranges from the same namespace, leading to undefined server behavior.

Clients MUST generate foreign identifiers to identify changes to objects in the local replica, as specified in section 3.3.5.1.3.

This mechanism is being serviced by two ROPs, **RopGetLocalReplicaIds** (section 2.2.3.2.4.7) and **RopSetLocalReplicaMidsetDeleted** (section 2.2.3.2.4.8).

To help compression of **IDSETs** and to alleviate fragmentation of the deleted item list, if a server maintains an **IDSET** for a folder, clients SHOULD assign consecutive IDs from the allocated range to messages within the same folder. This can be achieved by allocating a contiguous subset of allocated IDs to each folder.

Clients MUST report IDs assigned to objects in a client replica that were deleted without ever being uploaded through the **RopSynchronizationImportDeletes** ROP.

Clients MUST report ranges of server-allocated IDs, which will never be used for any messages in a folder, through the **RopSetLocalReplicaMidsetDeleted** ROP. For more details, see section 3.3.5.5.4.10.

3.3.5.1.2 Use Online Mode ROPs

In this approach, clients upload objects created in their local replica by using the regular, non-synchronization ROPs, such as **RopCreateFolder** or **RopCreateMessage**, as specified in [MS-OXCROPS], which makes servers assign internal identifiers (2) as usual. The following are the limitations of this mode:

- Clients do not have server-accepted identifiers for objects until after they are uploaded to a server.
- Clients do not control internal identifiers (2) assigned to objects and changes by a server.
- Clients cannot set values of special properties, such as **PidTagLastModificationTime** ([MS-OXPROPS] section 2.861).
- Clients are entirely responsible for updating the ICS state to prevent uploaded objects from being downloaded during a subsequent synchronization download operation.

3.3.5.1.3 Foreign Identifiers

Clients MUST generate foreign identifiers to identify changes to objects in the local replica. Foreign identifiers are represented as **XID** values, as specified in section 2.2.2.2, and MUST NOT have the same byte length as **GID** values ([MS-OXCDATA] section 2.2.1.3); that is, the number of bytes in the **LocalId** field that follows a **NamespaceGuid** in the **XID** structure MUST be different from the size of GLOBCNT, which is 6 bytes. At the same time, foreign identifiers that share the same **NamespaceGuid** MUST have the same length of the **LocalId** part.

Clients MUST create foreign identifiers within the **NamespaceGuids** they generated, and MUST NOT use any REPLGUIDs returned by a server for that purpose.

Foreign identifiers MUST have the same qualities as internal identifiers (2): they MUST be unique, MUST NOT ever be reused and MUST be guaranteed to increase for any new change, or use a different GUID. This is important for conflict detection, as specified in section 3.1.4.1.1.

3.3.5.2 Determining the Synchronization Scope

To be able to perform an ICS download of mailbox data, a client MUST subdivide all necessary synchronization work into smaller pieces, which clearly define boundaries of synchronization operations in the terms supported by the ICS protocol (see the **RopSynchronizationConfigure** ROP, as specified in section 2.2.3.2.1.1). Synchronization scope is determined by using the following variables:

- **Synchronization type** (hierarchy or contents), as indicated by the **SynchronizationType** enumeration of the **RopSynchronizationConfigure** ROP request.
- Folder within the mailbox, as indicated by the **InputServerObject** field of the **RopSynchronizationConfigure** ROP request.
- Restrictions on messages within the folder that are included in the scope (for content synchronization operations only), as indicated by the **RestrictionData** field of the **RopSynchronizationConfigure** ROP request.

Synchronization for each of the scopes can be performed independently. For each synchronization scope, a client MUST persist the corresponding ICS state and pass it along when configuring a synchronization operation, as specified in section 2.2.3. ICS state does not reflect the synchronization scope it belongs to. Therefore, a client MUST ensure that the ICS state it passes to a server corresponds to the synchronization scope that it was originally obtained for.

Examples of synchronization scopes include the following:

- Folder hierarchy that starts with folder X
- All contents of folder Z

- All unread messages in folder Y that were received within the last three days

Note that the set of messaging objects that are considered for ICS operation can be further limited with flags, such as Normal or FAI set in the **SynchronizationFlag** field of the **RopSynchronizationConfigure** ROP. However, these flags do not modify the synchronization scope; they just filter the output produced by an operation.

For example, consider the following ICS operation:

- IcsDownload(icsStateX, Normal | FAI) => (diffNormal diffFAI, icsStateZ)

This operation outputs differences for all the messages in a folder. Compare it with the following sequence of ICS operations:

1. IcsDownload(icsStateX, Normal) => (diffNormal, icsStateY)

2. IcsDownload(icsStateY, FAI) => (diffFAI, icsStateZ)

This sequence is correct and it will produce the same end result as the previous single step operation.

The following sequence, however, is incorrect, because it uses a different synchronization scope (by supplying a different value for the restriction field) for the same ICS state:

- IcsDownload(icsStateX, Normal | FAI, {**PidTagAssociated** (section 2.2.1.4) equals "FALSE"})

=> (diff1, icsStateA)

- IcsDownload(icsStateA, Normal | FAI, {**PidTagAssociated** equals "TRUE"})

=> (diff2, icsStateB)

As a result, this sequence will not yield the same result:

- diff1 will contain soft deletion notifications for any previously downloaded messaging objects mentioned in icsStateX. The **PidTagIdsetGiven** property (section 2.2.1.1.1), which does not have a **PidTagAssociated** property value equals "FALSE".

- diff2 will contain soft deletions for all messaging objects mentioned in icsStateA.**PidTagIdsetGiven**.

- icsStateB.**PidTagIdsetGiven** will only contain IDs of FAI messages.

3.3.5.3 Client Side Checkpointing

Checkpointing is a method of ICS state management that is used for upload and download operations to provide updated state information on an object by object basis. If a client has to abort synchronizations regularly, or if mitigating the effects of application or connection termination is a priority, the client can implement an ICS state checkpointing strategy. This is made possible by including the **SynchronizationExtraFlag Eid** and **CN** flags in the **ulExtra** field of the **RopSynchronizationConfigure** ROP request (section 2.2.3.2.1.1) that is made when initializing a download. Including the **Eid** and **CN** flags cause the server to send the **PidTagMid** property (section 2.2.1.2.1) and the **PidTagChangeNumber** property (section 2.2.1.2.3) in the ICS header for each messaging object. Using the **PidTagChangeNumber** property and the item's **PidTagMid** property, the ICS state can be maintained on the client by updating its **PidTagIdsetGiven** (section 2.2.1.1.1) and **PidTagCnsetSeen** (section 2.2.1.1.2) (or **PidTagCnsetSeenFAI** (section 2.2.1.1.3), for folder associated information (FAI) messages) properties by using the **PidTagMid** and

PidTagChangeNumber properties, respectively. Each of these properties is specified in section 2.2.1.1 or section 2.2.1.2.

If this updated ICS state is then persisted periodically, the client does not have to redownload all items in the event of a cancellation or other abnormal termination. It is recommended that the client persist the state downloaded at the end of the current (or a subsequent) download synchronization session after the download is complete. This is because the ICS state provided by the server is a much more efficient version than the version obtained by using checkpointing alone.

Note Checkpointing for synchronization download operations functions differently than checkpointing for synchronization upload operations. During a synchronization upload operation, the server returns checkpoint ICS states that are accurate to the time at which the checkpoint was requested. During a synchronization download operation, the server returns the initial ICS state until the download is complete, at which time it returns the final ICS state.<26>

3.3.5.4 Sending FastTransfer ROPs

3.3.5.4.1 Download

The following steps MUST be taken by a client to download copies of messaging objects from the server in FastTransfer mode:

1. Obtain a handle to a messaging object whose contents are requested or a handle to a messaging object that the client will download a copy of. To obtain a handle to a new messaging object, use the *OutputHandleIndex* parameter from the **RopCreateMessage** ROP ([MS-OXCROPS] section 2.2.6.2) response buffer. For more details about obtaining a handle to an existing messaging object, see [MS-OXCMSG] section 3.1.4.1.

2. Send the **RopFastTransferSourceCopy*** ROP request to create a FastTransfer download context on the server and define the parameters and the scope of the operation.

3. Optionally, send a **RopTellVersion** ROP request, if performing a server-to-client-to-server upload, as specified in section 3.3.5.4.2.1.

4. Iteratively send **RopFastTransferSourceGetBuffer** ROP requests on the FastTransfer context to retrieve the FastTransfer stream with serialized messaging objects.

5. Send a **RopRelease** ROP request to release the messaging object and FastTransfer context obtained in steps 1 and 2.

3.3.5.4.1.1 Sending a RopFastTransferSourceCopyTo Request

The object output in the **OutputServerObject** field MUST be released using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.4.1.2 Sending a RopFastTransferSourceCopyProperties Request

The object output in the **OutputServerObject** field MUST be released using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.4.1.3 Sending a RopFastTransferSourceCopyMessages Request

The object output in the **OutputServerObject** field MUST be released by using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.4.1.4 Sending a RopFastTransferSourceCopyFolder Request

The object output in the **OutputServerObject** field MUST be released using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.4.1.5 Sending a RopFastTransferSourceGetBuffer Request

The FastTransfer stream on download is read-only and non-seekable, and is usually generated on-the-fly. Once it is obtained, data cannot be requeried, unless the operation is reconfigured from the beginning. Even then, there is no guarantee that the content of the stream will be the same as during the previous attempt.

As streams can be very large, clients SHOULD decode portions of the FastTransfer stream as they arrive in the **RopFastTransferSourceGetBuffer** ROP response buffers.

To obtain all data output by an operation, the **RopFastTransferSourceGetBuffer** ROP MUST be sent iteratively, because the amount of data that can be passed in one RPC is limited by its maximum size. A client MUST stop sending this ROP on a download context as soon as it receives **TransferStatus Done** or **Error**.

Clients SHOULD set the value of the **BufferSize** field in the ROP request to a sentinel value of "0xBABE" to achieve maximum efficiency. If this field is not set to "0xBABE", then clients MUST pass a value equal to or greater than 15480<27> or MUST be prepared to increase this number in future requests if they passed a smaller value and the **RopBufferTooSmall** ROP was returned.

Clients MAY set the value of the **MaximumBufferSize** field to at least the size of the output RPC buffer to achieve maximum efficiency.

If the client receives a **ReturnValue** of **ServerBusy**, the client MUST wait at least the period of time specified in **BackoffTime** before retrying the ROP. The complete list of error codes is specified in [MS-OXCDATA] section 2.4.<28>

3.3.5.4.1.6 Sending a RopTellVersion Request

Clients MUST pass the version exactly as it was obtained from the **EcDoConnectEx** call results. For more details about the only application scenario for this ROP, server-to-client-to-server upload, see section 3.3.5.4.2.1.

If the client sends the **RopTellVersion** ROP, the request MUST be sent before the first **RopFastTransferSourceGetBuffer** or **RopFastTransferDestinationPutBuffer** ROP.

3.3.5.4.2 Upload

The following steps MUST be taken by a client to upload copies of messaging objects to the server in FastTransfer mode:

1. Obtain a handle to a messaging object, for which appending or replacing properties and/or subobjects is requested. To obtain a handle to a new messaging object, use the *OutputHandleIndex* parameter from the **RopCreateMessage** ROP ([MS-OXCROPS] section 2.2.6.2) response buffer. For more details about obtaining a handle to an existing messaging object, see [MS-OXCMSG] section 3.1.4.1.

2. Send the **RopFastTransferDestinationConfigure** ROP to create a FastTransfer upload context on the server and define the parameters of the operation.

3. Optionally, send the **RopTellVersion** ROP if performing a server-to-client-to-server upload, as specified in section 3.3.5.4.2.1.

4. Iteratively send the **RopFastTransferDestinationPutBuffer** ROP on the FastTransfer context to upload the FastTransfer stream with the serialized messaging objects.

5. Send the **RopRelease** ROP to release the messaging object and the FastTransfer context obtained in steps 1 and 2.

In step 4, if a client simply resends the stream that it is getting through the FastTransfer download, it can consider using an optimized server-to-client-to-server upload process, as specified in section 3.3.5.4.2.1.

3.3.5.4.2.1 Server-to-Client-to-Server Upload

To optimize copying messaging objects between two different mailboxes on two different servers by using FastTransfer upload paired with FastTransfer download, a client can specify the **ForUpload** flag in **SendOptions**, which instructs the source server to produce a FastTransfer stream that is optimized for the destination server.

Clients MUST NOT parse the FastTransfer stream produced by the source server, as it can contain optimizations and not adhere to the grammar specified in section 2.2.4.

Clients MUST use the following steps to execute server-to-client-to-server copying:

1. Send one of the **RopFastTransferSourceCopy*** ROP requests to server A to configure a FastTransfer download context, while setting the **ForUpload** flag in the **SendOptions** field.

2. Send the **RopFastTransferDestinationConfigure** ROP request to server B to configure a FastTransfer upload context.

3. Send the **RopTellVersion** ROP request on the FastTransfer download context with a version of server B.

4. Send the **RopTellVersion** ROP request on the FastTransfer upload context with a version of server A.

5. Iteratively send the **RopFastTransferSourceGetBuffer** ROP requests on the FastTransfer download context, followed by the **RopFastTransferDestinationPutBuffer** ROP requests on the FastTransfer upload context, until there is no more data.

6. Release both FastTransfer contexts.

3.3.5.4.2.2 Sending a RopFastTransferDestinationConfigure Request

The object output in the **OutputServerObject** field MUST be released using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.5 Sending Incremental Change Synchronization ROPs

The following figure shows the steps involved in ICS.

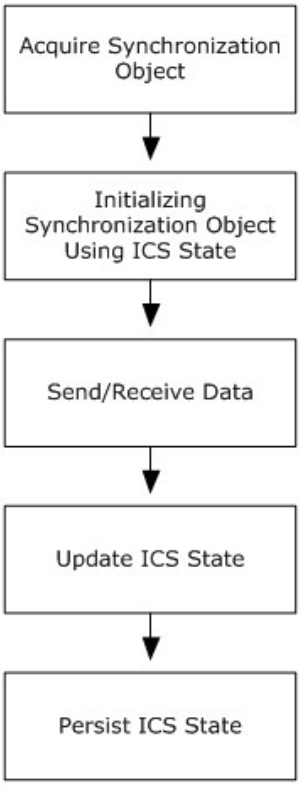


Figure 3: Steps in Incremental Change Synchronization

3.3.5.5.1 Download

The following steps MUST be taken by a client when downloading mailbox differences from a server:

1. Obtain a handle to a Folder object, for which synchronization is to be requested. For details about obtaining a folder handle, see [MS-OXCFOLD].
2. Send the **RopSynchronizationConfigure** ROP (section 2.2.3.2.1.1) request to create a synchronization download context on the server and define the parameters and the scope of the operation.
3. Send the **RopSynchronizationUploadStateStreamBegin** (section 2.2.3.2.2.1), **RopSynchronizationUploadStateStreamContinue** (section 2.2.3.2.2.2), and **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3) requests to upload the initial ICS state information to the synchronization context.
4. Iteratively send the **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.1.1.5) request on the synchronization download context to retrieve the FastTransfer stream of the mailbox differences and the final ICS state.
5. Persist the ICS state.

6. Send the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) request to release the Folder object and the synchronization download context obtained in steps 1 and 2.

3.3.5.5.1.1 Sending a RopSynchronizationConfigure Request

The client MUST upload the last remaining piece of configuration data, the initial ICS state, before it can request a FastTransfer stream that contains differences from the server.

The object output in **OutputServerObject** field MUST be released using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.5.2 Uploading State

After the synchronization context is acquired for an ICS download, the client MUST supply the initial ICS state, as specified in section 2.2.1.1, before executing any other ROPs on the synchronization context. For an ICS upload, the client SHOULD supply the initial ICS state; however, if no initial ICS state is provided, the server starts with an empty state. The following table summarizes the requirements for the ICS state properties being uploaded to different synchronization contexts.

ICS state property	Hierarchy download	Contents download	Hierarchy upload	Contents upload
PidTagIdsetGiven (section 2.2.1.1.1)	MUST	MUST	Not applicable	Not applicable
PidTagCnsetSeen (section 2.2.1.1.2)	MUST	MUST	SHOULD	SHOULD
PidTagCnsetSeenFAI (section 2.2.1.1.3)	Not applicable	MUST	Not applicable	SHOULD
PidTagCnsetRead (section 2.2.1.1.4)	Not applicable	MUST	Not applicable	SHOULD

Uploading the ICS state is done sequentially, property by property. The order in which properties are uploaded does not matter. The upload of each property MUST be initiated by sending the **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1) request, followed by one or more **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2) requests. The upload is finished with the **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3).

3.3.5.5.2.1 Sending a RopSynchronizationUploadStateStreamBegin Request

When the **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1) is sent to the server, no other property upload MUST be in progress for this synchronization context, and a property that is being specified in this ROP SHOULD NOT have been already uploaded into this synchronization context. This ROP MUST be followed by the **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2) or the **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3).

3.3.5.5.2.2 Sending a RopSynchronizationUploadStateStreamContinue Request

This ROP MUST be followed by the **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2) or the **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3). The upload MUST be initiated by sending the **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1).

Clients SHOULD skip this ROP if the size of the remaining data specified in the **StreamDataSize** field is 0.

3.3.5.5.2.3 Sending a RopSynchronizationUploadStateStreamEnd Request

The upload MUST be initiated by sending the **RopSynchronizationUploadStateStreamBegin** ROP request (section 2.2.3.2.2.1) followed by zero or more iterations of the **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2).

3.3.5.5.3 Downloading State

The client downloads the server ICS state using the **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1). For more details about the ICS state properties, see section 2.2.1.1. For more details about sending and receiving the **RopSynchronizationGetTransferState** ROP, see sections 3.3.5.5.3.1 and 3.2.5.5.3.1, respectively.

3.3.5.5.3.1 Sending a RopSynchronizationGetTransferState Request

Clients are only required to use the **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1) when performing synchronization uploads, as it is the only way to obtain the ICS state maintained on the synchronization upload context.

For synchronization downloads, the final ICS state is downloaded at the end of the FastTransfer stream, and this ROP can only be used to obtain the initial ICS state and final ICS state, as an alternative to using client-side checkpointing, as specified in section 3.3.5.3.<29>

The object output in **OutputServerObject** MUST be released by using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.5.4 Upload

The client uploads initial ICS state and downloads the final/checkpoint ICS state when doing synchronization upload. Clients can perform a synchronization upload without uploading the initial ICS state properties into a synchronization upload context, because the behavior of the **RopSynchronizationImport*** ROPs do not depend on the initial ICS state. In that case, a server can download the changes uploaded in this session during the subsequent ICS downloads.

The following figure shows the primary processes taking place during an upload operation. The sections that follow describe the details within the Send Data process.

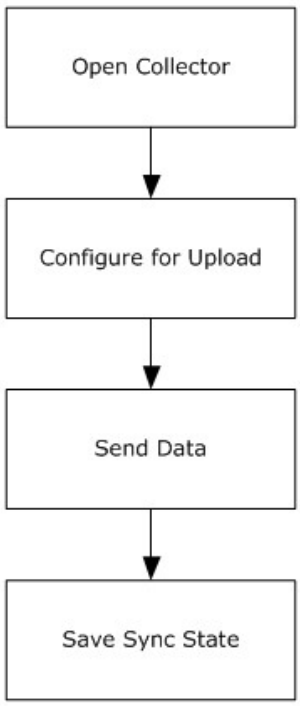


Figure 4: Upload operation

The following steps elaborate on the steps in the figure and MUST be taken by a client when uploading mailbox differences to a server:

1. Obtain a handle to the Folder object, as specified in [MS-OXCFOLD], that will be synchronized.
2. Send a **RopSynchronizationOpenCollector** ROP request (section 2.2.3.2.4.1) to create a synchronization upload context on the server and to define parameters and the scope of an operation.
3. SHOULD send the **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1), the **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2), and the **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3) request to upload the initial ICS state information to the synchronization context.
4. Upload changes, moves, and deletes of individual objects within the synchronized Folder object through **RopSynchronizationImport*** ROPs, while passing the synchronization upload context obtained in step 2. Uploading hierarchy changes is specified in section 3.3.5.5.4.1. Uploading content changes is specified in section 3.3.5.5.4.2.
5. SHOULD obtain the final ICS state by doing the following:

   - Acquire a separate FastTransfer download context for a checkpoint ICS state by using the **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1) and passing the synchronization upload context obtained in step 2 in the request buffer.

- Perform FastTransfer download step 4, as specified in section 2.2.3.1.1, on the FastTransfer download context acquired in the first bullet point.
- Release the FastTransfer download context obtained in the first bullet point.

6. Persist the ICS state.
7. Send the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) request to release the Folder object and the synchronization upload context obtained in steps 1 and 2.

The client can elect not to upload/download the ICS states in steps 3 and 5. For details on how that would affect responsibilities of the protocol roles, see section 3.3.5.5.4.

The following table lists the common return values from the **RopSynchronizationImport*** ROPs that clients SHOULD have special processing for.

Value	Description
Success	No error occurred, or a conflict has been resolved**.**
NoParentFolder	The parent folder never existed.
ObjectDeleted	An object or its parent folder has already been deleted.
IgnoreFailure	The change was ignored, as it has been superseded by another change.

The complete list of error codes is specified in [MS-OXCDATA] section 2.4.

3.3.5.5.4.1 Hierarchy Upload

The following sections specify best practices for uploading hierarchy modifications and deletions that were tracked by the client while the client was offline.

The Send Data process (as shown in Figure 4) is illustrated in Figure 5.

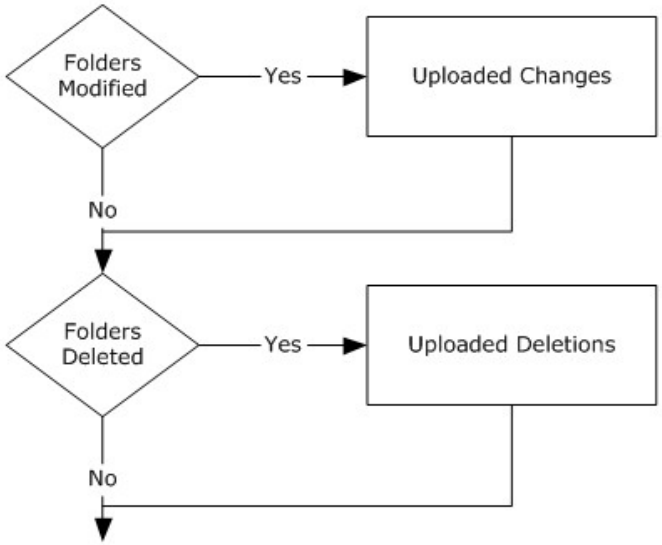


Figure 5: Send data process

When uploading hierarchy differences, the client sends the following ROP requests:

- **RopSynchronizationImportHierarchyChange** (section 2.2.3.2.4.3)
- **RopSynchronizationImportDeletes** (section 2.2.3.2.4.5)

3.3.5.5.4.1.1 Uploading Hierarchy Changes

New and modified folders are uploaded in the same manner. A client MUST collect all properties that are stored on the local replica, and use the synchronization upload context and the **RopSynchronizationImportHierarchyChange** ROP (section 2.2.3.2.4.3), to transmit this information to the server. When public folders are uploaded, the synchronization upload context is opened by using the folder that is being synchronized. When this happens, a **PidTagParentSourceKey** property (section 2.2.1.2.6) with a count of bytes of 0 (zero) is used to denote that the folder properties belong to the folder from which the synchronization upload context was opened. A move of a folder from one parent to another is modeled as a modification of a folder, where the value of **PidTagParentSourceKey** of the folder changes to reflect the new parent.

There is no mechanism to represent conflicts on hierarchy. As such, the server MUST apply "last writer wins" semantics to hierarchy change uploads. The last writer wins algorithm is specified in section 3.1.4.1.2.2.

Clients SHOULD ignore the following errors, which indicate that the server did not apply the changes:

Value	Description
ObjectDeleted	An object or its parent folder has already been deleted.
IgnoreFailure	The change was ignored, as it has been superseded by another change.

The complete list of error codes is specified in [MS-OXCDATA] section 2.4.

3.3.5.5.4.1.2 Uploading Hierarchy Deletions

Folder deletions are performed by transmitting the **PidTagSourceKey** property (section 2.2.1.2.5) for the folder to be removed by using the synchronization upload context and the **RopSynchronizationImportDeletes** ROP (section 3.3.5.5.4.7).

The server MUST stop processing deletion operations upon encountering the first error; therefore, a client MUST be prepared to retry the operations on a newly initialized synchronization upload context if a failure is encountered.

Errors can occur during bulk operations that require additional effort to disambiguate and rectify. These errors can occur more often when the user is not the owner of the mailbox.

3.3.5.5.4.2 Content Upload

The following sections specify best practices for uploading content modifications, read/unread state changes, deletions, and move operations that were performed while the client was offline. The following figure shows this process.

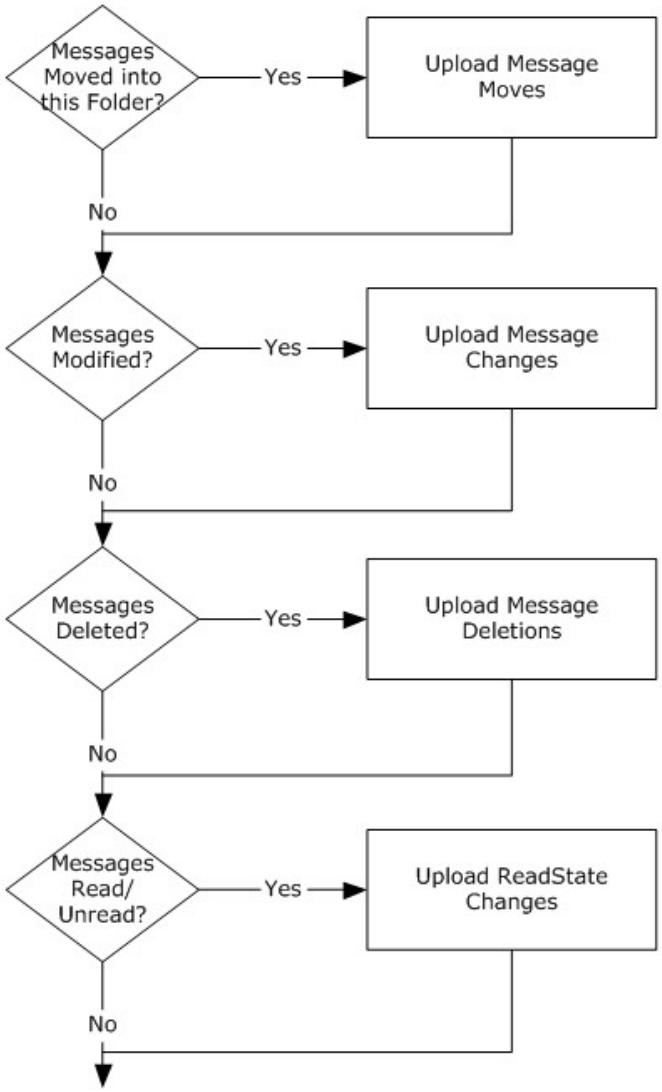


Figure 6: Messages best practices

When uploading content differences, the client can send any combination of the following ROP requests:

- **RopSynchronizationImportMessageChange** (section 2.2.3.2.4.2). Imports new messages or changes to existing messages.
- **RopSynchronizationImportMessageMove** (section 2.2.3.2.4.4). Communicates the movement of messages between folders within the same mailbox.
- **RopSynchronizationImportDeletes** (section 2.2.3.2.4.5). Imports deletions of messages.

- **RopSynchronizationImportReadStateChanges** (section 2.2.3.2.4.6). Imports changes to the read state of messages.

These ROPs do not have to be sent in any specific order and can be mixed together. For example, all the deletions do not have to be uploaded before all the message moves, and all the message changes do not have to be uploaded before all the deletions. For more information about best practices for ordering different types of upload and download operations, see [MS-OXCSYNC] section 3.1.5.2.

3.3.5.5.4.2.1 Uploading Moves

3.3.5.5.4.2.1.1 Moves and Modifications

Message moves MUST be performed using a synchronization upload context of the folder to which the message was moved (the destination folder). To synchronize a message move, use the **RopSynchronizationImportMessageMove** ROP (section 2.2.3.2.4.4). The client specifies the source folder information.

3.3.5.5.4.2.1.2 Avoiding Duplicate Uploads

When a client that is using a local replica sends a message by using a server with a major version of less than eight (8), a new message is created in a server folder, the contents of the local message are copied into the server message, and the message is submitted (on the first upload). After the submission is complete, the item is moved into the **Sent Items folder** in the local replica. Sometime later, the Sent Items folder is synchronized. The item will then be uploaded as it is a new item in the replica (on the second upload) due to the fact that the folder to which the message was originally submitted is not within the user's mailbox (and therefore not part of the replica).

Servers with a major version of 8 and above implement the following alternative method to diminish the impact of the second upload. For details about determining server version, see section 1.7. Because the client assigns server IDs to all items it creates in the local replica, the message in the **Outbox folder** has a valid server ID. When the client is uploading the message for sending, it adds an additional property to the message, **PidTagTargetEntryId** ([MS-OXOMSG] section 2.2.1.66). When a version 8 (or later) server observes this property, it places a mirror copy of the message in the user's server Outbox and give it the ID specified in the **PidTagTargetEntryId** property. In this way, when the client synchronizes the folder holding sent items, it can be synchronized as a move (from the Outbox folder to the folder holding sent items) as opposed to a new item (in the folder holding sent items).

3.3.5.5.4.2.2 Uploading Modifications

The following sections specify the processes for uploading message modifications. Conflict detection is defined in section 3.1.4.1.1

When a message has been moved and modified in the offline store, a client MUST apply modifications after moving the message, as specified in section 3.3.5.5.4.2.1.1.

3.3.5.5.4.2.2.1 Full Item Upload

Message changes are uploaded by using the **RopSynchronizationImportMessageChange** ROP (section 2.2.3.2.4.2), followed by copying all properties to the message by using several messaging ROPs, as specified in [MS-OXCMSG], and persisting the changes using the **RopSaveChangesMessage** ROP, as specified in [MS-OXCMSG] section 2.2.3.3.

Clients SHOULD ignore the following errors (returned from the **RopSynchronizationImportMessageChange** ROP and the **RopSaveChangesMessage** ROP, or both), which indicate that the server did not apply the changes.

Value	Description
ObjectDeleted	An object or its parent folder has already been deleted.
IgnoreFailure	The change was ignored, as it has been superseded by another change.

The complete list of error codes is specified in [MS-OXCDATA] section 2.3.

A client SHOULD have a strategy to deal with non-transient errors to prevent them from occurring in subsequent synchronization attempts. That strategy can be to move items that fail to upload to a local folder that is not synchronized. In addition, for non-new items, the client can attempt to bring down the previous version of the item in order to get a good version of the item back in the user's offline store.

When detecting conflicting changes, a server MUST perform conflict resolution as defined in section 2.2.1.6.

3.3.5.5.4.2.2.2 Partial Item Upload

To improve wire efficiency, a client SHOULD track offline changes in such a way that these can be uploaded individually without having to transmit the full item. One strategy is for a client to apply the changes by using standard Message object and Attachment object protocol calls, as specified in [MS-OXCMSG], such as the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3), the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6), the **RopSetReadFlags** ROP ([MS-OXCROPS] section 2.2.6.10), and the **RopDeleteProperties** ROP ([MS-OXCROPS] section 2.2.8.8), to apply changes directly to the server replica. Then, the client uses the **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) together with the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3) to get the new PCL and **CN** values for the message, as specified in sections 2.2.2.3 and 2.2.2.1, respectively. The client SHOULD then modify the message PCL, message **CN**, and ICS state on the client in a way that would prevent the item from being downloaded, as specified in section 3.1.4.1.

Using this mechanism prevents the client from controlling certain internal properties, such as **PidTagLastModificationTime** ([MS-OXPROPS] section 2.861), as this are be controlled by the server. This is different than full item upload, which accepts the client's value of this property.

3.3.5.5.4.2.2.3 Conflict Resolution

A client SHOULD avoid conflicts by detecting them and trying to run logic to resolve the conflict. One strategy would be for a client to do this by using standard message and folder ROPs, such as the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3), the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6), the **RopSetReadFlags** ROP ([MS-OXCROPS] section 2.2.6.10), and the **RopDeleteProperties** ROP ([MS-OXCROPS] section 2.2.8.8), to apply changes to the server replica, therefore making it a nonconflicting version. The client then modifies the PCL and ICS state in a way that would either prevent the item from being downloaded (local wins), or force (resolved or server wins) the item to be downloaded, as specified in section 3.1.4.1.

Using this mechanism prevents the client from controlling certain internal properties, such as **PidTagLastModificationTime** ([MS-OXPROPS] section 2.861), as these properties are be controlled by the server. This is different than full item upload which accepts the client's value of this property.

3.3.5.5.4.2.3 Uploading Deletes

Message deletions are performed by transmitting the **PidTagSourceKey** property (section 2.2.1.2.5) for the messages to be removed by using the **RopSynchronizationImportDeletes** ROP (section 2.2.3.2.4.5) on a synchronization upload context of the folder that contains those messages.

A client SHOULD batch **PidTagSourceKey** entries for multiple messages to improve wire efficiency.

The server MUST stop processing deletion operations upon encountering the first error that is not NotFound, as specified in [MS-OXCDATA] section 2.4, so a client MUST be prepared to retry the operations on a newly initialized synchronization upload context if a failure is encountered.

Errors can occur during bulk operations that require additional effort to disambiguate and rectify. These errors occur more often when the user is not the owner of the mailbox.

3.3.5.5.4.2.4 Uploading Read/Unread State Changes

Message read/unread state is uploaded by transmitting one structure per message, which changes the status by using the synchronization upload context and the **RopSynchronizationImportReadStateChanges** ROP (section 2.2.3.2.4.6).

For each message that has a read state change, the client MUST specify the value of the **PidTagSourceKey** property (section 2.2.1.2.5) and flags, indicating the updated read state.

Errors can occur during bulk operations that require additional effort to disambiguate and rectify. These errors occur more often when the user is not the owner of the mailbox.

3.3.5.5.4.3 Sending a RopSynchronizationOpenCollector Request

A client SHOULD upload the initial ICS state, as specified in section 2.2.3.2.2, into the synchronization context returned in the **RopSynchronizationOpenCollector** ROP (section 2.2.3.2.4.1) response prior to using any RopSynchronizationImport* ROPs. The client can elect not to upload the initial ICS state. For details about how that would affect responsibilities of the protocol roles, see section 3.3.5.5.4.

Be sure to update the stored **PidTagIdsetGiven** property (section 2.2.1.1.1) value with internal identifiers (2) of the objects that were imported into the server replica. These identifiers either are returned in the **RopSynchronizationImport*** ROP responses or can be extracted from **GID** values ([MS-OXCDATA] section 2.2.1.3) sent as the input **PidTagSourceKey** property (section 2.2.1.2.5) values.

The object output in the **OutputServerObject** field MUST be released by using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.5.4.4 Sending a RopSynchronizationImportMessageChange Request

When uploading new messages, clients SHOULD add their **MID** values ([MS-OXCDATA] section 2.2.1.2) to the **PidTagIdsetGiven** property (section 2.2.1.1.1) value upon successful completion of this ROP.

Note that because a server returns an empty message from the **RopSynchronizationImportMessageChange** ROP ([MS-OXCROPS] section 2.2.13.2), even when uploading changes to an existing message, this ROP can only be used to perform upload of full message changes or new messages. In order for the client to upload partial message changes, it SHOULD take them outside the synchronization upload operation, by initiating an upload by using

the **RopOpenMessage** ROP ([MS-OXCROPS] section 2.2.6.1) followed by other ROPs discussed in [MS-OXCMSG], such as the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) and the **RopModifyRecipients** ROP ([MS-OXCROPS] section 2.2.6.5). However, these ROPs do not let the client set values to any of the properties that the **RopSynchronizationImportMessageChange** ROP accepts.

The **RopSynchronizationImportMessageChange** ROP returns the handle of a Message object, which the client MUST populate with the contents of the message. The client populates the Message object by sending the **ROPSetProperties** ROP, the **ROPCreateAttachment** ROP ([MS-OXCROPS] section 2.2.6.13), and other ROPs required to populate the message contents, as specified in [MS-OXCMSG] section 3.1.4, followed by the **ROPSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3).

The object output in the **OutputServerObject** field MUST be released by using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) as soon as the client no longer requires it.

3.3.5.5.4.5 Sending a RopSynchronizationImportHierarchyChange Request

When uploading new folders, clients SHOULD update the ICS state that corresponds to the chosen synchronization scope by adding **FID** values ([MS-OXCDATA] section 2.2.1.1) of new folders to the **PidTagIdsetGiven** property (section 2.2.1.1.1) upon successful completion of this ROP.

Changes to parent folders MUST be made before changes to child folders. For example, the client cannot send the **RopSynchronizationImportHierarchyChange** ROP (section 2.2.3.2.4.3) with a subfolder change before informing the server of the existence of the parent folder.

To move a folder to a different subfolder within the same private mailbox, the client MUST pass the **PidTagSourceKey** property (section 2.2.1.2.5) value of a destination parent folder in the **PidTagParentSourceKey** property (section 2.2.1.2.6) value in the HierarchyValues field while passing the **PidTagSourceKey** value of the folder being moved in the **PidTagSourceKey** property. Moving folders within a public mailbox is not supported.

3.3.5.5.4.6 Sending a RopSynchronizationImportMessageMove Request

When uploading new messages, clients SHOULD update the ICS state of the source folder by removing the **MID** values ([MS-OXCDATA] section 2.2.1.2) of moved messages from its **PidTagIdsetGiven** property (section 2.2.1.1.1). Otherwise, the client MUST be prepared to receive deletion notifications for these messages in the source folder during the next ICS download.

Clients MUST only pass folders from private mailboxes in **InputServerObject**.

3.3.5.5.4.7 Sending a RopSynchronizationImportDeletes Request

Clients SHOULD update the ICS state of the chosen synchronization scope by removing internal identifiers (2) of deleted objects from its **PidTagIdsetGiven** property (section 2.2.1.1.1). Otherwise, the client will receive deletion notifications for these messages during the next ICS download.

Clients SHOULD expect this ROP to fail if deletion of any of the objects passed in the request buffer fail, except for the common cases specified in section 2.2.3.2.4.5. The possibility of a failure is higher when the user has lower privileges to a mailbox—this is especially a consideration for delegate and public folder access. It is recommended that clients that use this ROP have a strategy to retry this operation, which can be a combination of the following steps:

1. Retry the ROP with the same arguments on a new synchronization upload context.

2. Retry the ROP, passing one ID at a time.

3. Retry the ROP by using online mode ROPs, like **RopDeleteFolder** and **RopDeleteMessages**, as specified in [MS-OXCFOLD] section 2.2.3 and [MS-OXCFOLD] section 2.2.11, respectively.

4. Perform the ICS download, resolving server changes against their own pending synchronization upload context.

5. Skip an object and undo the operation in the local replica.

3.3.5.5.4.8 Sending a RopSynchronizationImportReadStateChanges Request

Clients SHOULD expect this ROP to fail if any read state changes on the objects passed in the request buffer fail. The possibility of a failure is higher when the user has lower privileges to a mailbox; this is especially a consideration for delegate and public folder access. Clients that use this ROP SHOULD have a strategy to retry this operation, which can be a combination of the following steps:

- Retry the ROP with the same arguments on a new synchronization upload context.
- Retry the ROP, passing one ID at a time.
- Retry the ROP by using online mode ROPs, such as the **RopSetMessageReadFlag** ROP ([MS-OXCROPS] section 2.2.6.11).
- Perform the ICS download, resolving server changes against their own pending synchronization upload context.
- Skip an object and undo the operation in the local replica.

3.3.5.5.4.9 Sending a RopGetLocalReplicaIds Request

Clients SHOULD NOT allocate another batch of IDs until the one they allocated before is used up. Allocating IDs in batches of moderate size, between 0x00000200 and 0x0000FFFF, is recommended. Note that servers are responsible for enforcing restrictions on the number of IDs that can be allocated at one time.

The client can reconstruct all allocated **GID** values ([MS-OXCDATA] section 2.2.1.3) by combining the returned REPLGUID with any GLOBCNT values from the [GlobalCount, GlobalCount + IdCount – 1] range.

The client SHOULD use the obtained IDs whenever creating new folders or new messages in any folder within its local replica. For more details about how clients can assign identifiers to objects created in a local replica, see section 3.3.5.1.

3.3.5.5.4.10 Sending a RopSetLocalReplicaMidsetDeleted Request

The following example shows a possible implementation of the client with regards to assignment of server-allocated IDs, as specified in section 3.3.5.1.1, to objects in a local replica. Clients do not have to follow the example specified in this section; it is only used to show the applicability of the **RopSetLocalReplicaMidsetDeleted** ROP (section 2.2.3.2.4.8):

1. Initially, a client has no server-allocated IDs that it can assign to objects that are created when working offline, so it is required to ask the server to allocate a block of IDs by sending the **RopGetLocalReplicaIds** ROP (section 2.2.3.2.4.7). The server responds with a block of IDs that the client stores in a local replica.

2. The client requires the server-allocated ID whenever it has to create a message in a folder in a local replica. For that purpose, the client associates a range of IDs previously allocated with the **RopGetLocalReplicaIds** ROP with a folder, so that IDs from that range can be used for new or moved items in that folder.

3. If a folder does not have a range of server-allocated IDs associated with it, because the previous range was depleted (say, [A; B]), the client would have to allocate another range (say, [C; D]) from the block obtained in step 1 and associate it with that folder.

4. After a new range [C; D] is associated with a folder, the client knows that all ids in [B+1; C-1] will never be used in that folder, because they have already been associated with other folders. Therefore, the client can send the **RopSetLocalReplicaMidsetDeleted** ROP for that folder with the [B+1; C-1] range.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

4.1 Hierarchy Synchronization

4.1.1 Add or Modify a Folder

The following is a sample conversation with the server. The user has previously created or modified a folder on both the client and the server and has just connected to the server to synchronize the changes.

1. **RopLogon** ROP ([MS-OXCROPS] section 2.2.3.1) – The server returns the ID of the **interpersonal messaging subtree** folder in this call.
2. **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1) – Open the interpersonal messaging subtree folder.
3. **RopSynchronizationOpenCollector** ROP (section 2.2.3.2.4.1) – Open the hierarchy synchronization upload context by using the handle of the interpersonal messaging subtree folder.
4. **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1) – Upload the ICS state property **PidTagCnsetSeen** (section 2.2.1.1.2) by using the synchronization upload context.
5. **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.
6. **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.
7. **RopSynchronizationImportHierarchyChange** ROP (section 2.2.3.2.4.3) – Send the folder properties by using the synchronization upload context.
8. **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1) – Get the updated ICS state by using the synchronization upload context. This call will return a handle to a synchronization download context.
9. **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.1.1.5) – Retrieve the ICS state data by using the synchronization download context.
10. **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) – Release the synchronization download context.
11. **RopRelease** – Release the synchronization upload context.
12. **RopSynchronizationConfigure** ROP (section 2.2.3.2.1.1) – Open the hierarchy synchronization download context by using the handle of the interpersonal messaging subtree folder.
13. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagIdsetGiven** (section 2.2.1.1.1) by using the synchronization download context.
14. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.
15. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

16. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

17. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

18. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

19. **RopFastTransferSourceGetBuffer** ROP – Receive the folder properties and updated ICS state by using the synchronization download context.

20. **RopRelease** ROP – Release the synchronization download context.

21. **RopRelease** ROP – Release the interpersonal messaging subtree folder.

22. **RopRelease** ROP – Release the store.

4.1.2 Delete a Folder

The following is a sample conversation with the server. The user has previously deleted a folder on both the client and the server, and has just connected to the server to synchronize the changes.

1. **RopLogon** ROP ([MS-OXCROPS] section 2.2.3.1) – The server returns the ID of the interpersonal messaging subtree folder in this call.

2. **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1) – Open the interpersonal messaging subtree folder.

3. **RopSynchronizationOpenCollector** ROP (section 2.2.3.2.4.1) – Open the hierarchy synchronization upload context by using the handle of the interpersonal messaging subtree folder.

4. **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1) – Upload the ICS state property **PidTagCnsetSeen** (section 2.2.1.1.2) by using the synchronization upload context.

5. **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

6. **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.2) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

7. **RopSynchronizationImportDeletes** ROP (section 2.2.3.2.4.5) – Send the information about the deleted folder by using the synchronization upload context.

8. **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1) – Get the updated ICS state by using the synchronization upload context. This call will return a handle to a synchronization download context.

9. **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.1.1.5) – Retrieve the ICS state data by using the synchronization download context.

10. **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) – Release the synchronization download context.

11. **RopRelease** ROP – Release the synchronization upload context.

12. **RopSynchronizationConfigure** ROP (section 2.2.3.2.1.1) – Open the hierarchy synchronization download context by using the handle of the interpersonal messaging subtree folder.

13. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagIdsetGiven** (section 2.2.1.1.1) by using the synchronization download context.

14. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

15. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

16. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

17. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

18. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

19. **RopFastTransferSourceGetBuffer** ROP – Receive the information about the deleted folder and updated ICS state by using the synchronization download context.

20. **RopRelease** ROP – Release the synchronization download context.

21. **RopRelease** ROP – Release the interpersonal messaging subtree folder.

22. **RopRelease** ROP – Release the store.

4.2 Sample Message Synchronization Upload

4.2.1 Add or Modify a Message

The following is a sample conversation with the server. The user has previously created or modified a message on both the client and the server, and has just connected to the server to synchronize the changes.

1. **RopLogon** ROP ([MS-OXCROPS] section 2.2.3.1) – Open the store.

2. **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1) – Open the folder being synchronized.

3. **RopSynchronizationOpenCollector** ROP (section 2.2.3.2.4.1) – Open the content synchronization upload context by using the handle of the folder being synchronized.

4. **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.4.1) – Upload the ICS state property **PidTagCnsetSeen** (section 2.2.1.1.2) by using the synchronization upload context.

5. **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.4.1) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

6. **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.4.1) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

7. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** (section 2.2.1.1.3) by using the synchronization upload context.

8. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization upload context.

9. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization upload context.

10. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetRead** (section 2.2.1.1.4) by using the synchronization upload context.

11. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization upload context.

12. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization upload context.

13. **RopSynchronizationImportMessageChange** ROP (section 2.2.3.2.4.2) – Acquire a Message object (with a specified ID) by using the synchronization upload context. If the message does not yet exist, it will be created. This call will return a handle to a Message object.

14. **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) – Set the message properties by using the message handle.

15. **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) – Save the message by using the message handle.

16. **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) – Release the message.

17. **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1) – Get the updated ICS state by using the upload context. This call will return a handle to a synchronization download context.

18. **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.2.3.1) – Retrieve the ICS state data by using the synchronization download context.

19. **RopRelease** ROP – Release the synchronization download context.

20. **RopRelease** ROP – Release the synchronization upload context.

21. **RopSynchronizationConfigure** ROP (section 2.2.3.2.3.1) – Open the content synchronization download context by using the handle of the folder being synchronized.

22. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagIdsetGiven** (section 2.2.1.1.1) by using the synchronization download context.

23. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

24. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

25. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

26. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

27. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

28. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

29. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

30. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

31. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

32. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

33. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

34. **RopFastTransferSourceGetBuffer** ROP – Receive the folder properties and updated ICS state by using the synchronization download context.

35. **RopRelease** ROP – Release the synchronization download context.

36. **RopRelease** ROP – Release the folder.

37. **RopRelease** ROP – Release the store.

4.2.2 Delete a Message

The following is a sample conversation with the server. The user has previously deleted a message on both the client and the server and has just connected to the server to synchronize the changes.

1. **RopLogon** ROP ([MS-OXCROPS] section 2.2.3.1) – Open the store.

2. **RopOpenFolder** ROP ([MS-OXCFOLD] section 2.2.1) – Open the folder that is being synchronized.

3. **RopSynchronizationOpenCollector** ROP (section 2.2.3.2.4.1) – Open the content synchronization upload context by using the handle of the folder that is being synchronized.

4. **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1) – Upload the ICS state property **PidTagCnsetSeen** (section 2.2.3.2.1.1) by using the synchronization upload context.

5. **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

6. **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

7. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** (section 2.2.1.1.3) by using the synchronization upload context.

8. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization upload context.

9. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization upload context.

10. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetRead** (section 2.2.1.1.4) by using the synchronization upload context.

11. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization upload context.

12. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization upload context.

13. **RopSynchronizationImportDeletes** ROP (section 2.2.3.2.4.5) – Send the information about the deleted message by using the synchronization upload context.

14. **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1) – Get the updated ICS state by using the synchronization upload context. This call will return a handle to a synchronization download context.

15. **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.1.1.5) – Retrieve the ICS state data by using the synchronization download context.

16. **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) – Release the synchronization download context.

17. **RopRelease** ROP – Release the synchronization upload context.

18. **RopSynchronizationConfigure** (section 2.2.3.2.1.1) – Open the content synchronization download context by using the handle of the folder that is being synchronized.

19. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagIdsetGiven** (section 2.2.1.1.1) by using the synchronization download context.

20. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

21. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

22. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

23. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

24. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

25. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

26. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

27. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

28. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

29. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

30. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

31. **RopFastTransferSourceGetBuffer** ROP – Receive the information about the deleted message and updated ICS state by using the synchronization download context.

32. **RopRelease** ROP – Release the synchronization download context.

33. **RopRelease** ROP – Release the folder.

34. **RopRelease** ROP – Release the store.

4.3 Sample Partial Item

4.3.1 Upload

The following is a sample conversation with the server. The user has previously modified a message on the client and has just connected to the server to synchronize the change. This partial upload will occur without using RopSynchronizationImportMessageChange.

1. **RopLogon** ROP ([MS-OXCSTOR] section 2.2.1.1) – Open the store.

2. **RopOpenFolder** ROP ([MS-OXCFOLD] section 2.2.1) – Open the folder being synchronized.

3. **RopSynchronizationOpenCollector** ROP (section 2.2.3.2.4.1) – Open the content synchronization upload context by using the handle of the folder being synchronized.

4. **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1) – Upload the ICS state property **PidTagCnsetSeen** ROP (section 2.2.3.1.1.5) by using the synchronization upload context.

5. **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.1) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

6. **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3) – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization upload context.

7. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** (section 2.2.1.1.3) by using the synchronization upload context.

8. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization upload context.

9. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization upload context.

10. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetRead** (section 2.2.1.1.4) by using the synchronization upload context.

11. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization upload context.

12. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization upload context.

13. **RopOpenMessage** ROP ([MS-OXCMSG] section 2.2.3.1) – Acquire a Message object with a specified ID. This call will return a handle to a Message object.

14. **RopGetPropertiesSpecific** ROP ([MS-OXCPRPT] section 2.2.2) – Get the values of **PidTagPredecessorChangeList** (section 2.2.1.2.8) and **PidTagChangeKey** (section 2.2.1.2.7) by using the message handle.

15. **RopDeletePropertiesNoReplicate** ROP ([MS-OXCPRPT] section 2.2.8) – Delete properties that were deleted on the local message.

16. **RopSetProperties** ROP ([MS-OXCPRPT] section 2.2.5) – Set new/updated message properties by using the message handle. In addition, also set an updated value of **PidTagPredecessorChangeList** to avoid having this change redownloaded to the client.

17. **RopSaveChangesMessage** ROP ([MS-OXCMSG] section 2.2.3.3) – Save the message by using the message handle.

18. **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) – Release the message.

19. **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1) – Get the updated ICS state by using the synchronization upload context. This call will return a handle to a synchronization download context.

20. **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.1.1.5) – Retrieve the ICS state data by using the synchronization download context.

21. **RopRelease** ROP – Release the synchronization download context.

22. **RopRelease** ROP – Release the synchronization upload context.

23. **RopRelease** ROP – Release the folder.

24. **RopRelease** ROP – Release the store.

4.3.2 Download

The following is a sample conversation with the server. The user has previously modified a message on the server and has just connected to the server to synchronize the changes by using the partial item download flag.

1. **RopLogon** ROP ([MS-OXCSTOR] section 2.2.1.1) – Open the store.

2. **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1) – Open the folder being synchronized.

3. **RopSynchronizationConfigure** ROP (section 2.2.3.2.1.1) – Open the content synchronization download context by using the handle of the folder being synchronized. Specify the **SendOptions PartialItem** (section 2.2.3.1.1.1.2) flag when using this ROP.

4. **RopSynchronizationUploadStateStreamBegin** ROP (section 2.2.3.2.2.1) – Upload the ICS state property **PidTagIdsetGiven** (section 2.2.1.1.1) by using the synchronization download context.

5. **RopSynchronizationUploadStateStreamContinue** ROP (section 2.2.3.2.2.2) – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

6. **RopSynchronizationUploadStateStreamEnd** ROP (section 2.2.3.2.2.3) – Upload the ICS state property **PidTagIdsetGiven** by using the synchronization download context.

7. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeen** (section 2.2.1.1.2) by using the synchronization download context.

8. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

9. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeen** by using the synchronization download context.

10. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** (section 2.2.1.1.3) by using the synchronization download context.

11. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

12. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetSeenFAI** by using the synchronization download context.

13. **RopSynchronizationUploadStateStreamBegin** ROP – Upload the ICS state property **PidTagCnsetRead** (section 2.2.1.1.4) by using the synchronization download context.

14. **RopSynchronizationUploadStateStreamContinue** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

15. **RopSynchronizationUploadStateStreamEnd** ROP – Upload the ICS state property **PidTagCnsetRead** by using the synchronization download context.

16. **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.2.2.3) – Receive the folder properties and updated ICS state by using the synchronization download context. These buffers will contain partial items as appropriate.

17. **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3) – Release the synchronization download context.

18. **RopRelease** ROP – Release the folder.

19. **RopRelease** ROP – Release the store.

4.4 IDSET Serialization

To efficiently transfer large numbers of **MID** values ([MS-OXCDATA] section 2.2.1.2) and **FID** values ([MS-OXCDATA] section 2.2.1.1) that identify changed or new messaging objects, the **MID** values and the **FID** values are serialized into an **IDSET** for transfer across the wire. The following example shows how to format and serialize an **IDSET**. Because of the variability of the **GLOBSET** encoding commands that are used within the serialization of an **IDSET**, an **IDSET** can be encoded in many different ways. There is no single correct way to encode a **GLOBSET** as long as the **GLOBSET**, when decoded, contains the same set of GLOBCNT values. The following is just one way to encode an **IDSET**.

This example uses an **IDSET** with following four **MID** values:

IDSET

	Value	REPLID	GLOBCNT
MID1	01 00 00 00 00 00 00 05	0001	000000000005

	Value	REPLID	GLOBCNT
MID2	01 00 00 00 00 00 00 06	0001	000000000006
MID3	01 00 00 00 00 00 00 10	0001	000000000010
MID4	02 00 00 00 00 00 00 09	0002	000000000009

The **IDSET** has to be properly formatted for serializations. For more details about how to format an **IDSET**, see section 3.1.5.5.1.

The following diagram represents how the **IDSET** has to be arranged for serialization. The individual ID values have been arranged by REPLID and the GLOBCNT values have been reduced to a **GLOBSET** for each REPLID. Within the **GLOBSET**, the GLOBCNT values are placed into contiguous ranges.

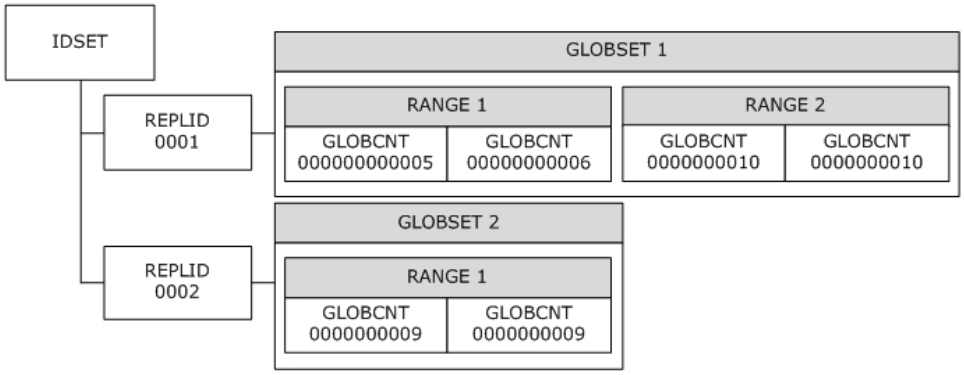


Figure 7: Arranging the IDSET for serialization

This example serializes the **IDSET** by using the REPLID format. For more details about the different serialization formats of an **IDSET**, see section 2.2.2.4.

For each REPLID/**GLOBSET** pair, the REPLID has to be added to the serialization buffer before the encoded **GLOBSET**. They have to be ordered based on the REPLID value where they are ordered from lowest to highest value.

The serialization buffer will resemble the following:

Serialization Buffer
01 00 <encoded GLOBSET 1> 02 00 <encoded GLOBSET 2>

GLOBSET 1 contains four GLOBCNT values; two in each GLOBCNT range. The encoding has to be performed based on the same order in which they are arranged in GLOBCNT ranges: from lowest to highest value. The following table is a list of all the GLOBCNT values in the order in which they have to be encoded.

#	GLOBCNT
1	00 00 00 00 00 05
2	00 00 00 00 00 06
3	00 00 00 00 00 10
4	00 00 00 00 00 10

Because all values have the same five bytes in common, the **Push** command can be used to push the five common bytes onto the common byte stack.

Current Encoding Buffer
05 00 00 00 00 00

Low and high GLOBCNT values in all ranges have to be evaluated in pairs. Because value 1 is close to value 2, it is possible to continue to evaluate subsequent ranges of GLOBCNT values to see if the **Bitmask** command can be used. However, values 3 and 4 are not close enough to value 1 to use the **Bitmask** command. Because only one GLOBCNT range will be put into a **Bitmask** command, either the **Bitmask** command or the **Range** command could be used. Because they both will occupy the same number of bytes in the encoded buffer, whether to use a **Bitmask** or **Range** command is an implementation decision. Both methods when decoded will result in the same GLOBCNT range. In this example, the **Range** command is used with the values 0x05 and 0x06 following it.

Current Encoding Buffer
05 00 00 00 00 00 **52 05 06**

This results in encodings to generate GLOBCNT values 1 and 2 if decoded. For GLOBCNT value 3 and 4, because they both have five bytes in common that are already in the common byte stack, no **Pop** or **Push** command has to be used. Because values 3 and 4 are close in value (in this particular case, they are identical), the **Bitmask** command could be used. Because there are no more GLOBCNT ranges to encode, the **Bitmask** command will only contain one range that takes 3 bytes of encoding. This is the same size a **Range** command would be to encode the same range. However, because the range is a singleton, it is more efficient to use the **Push** command to fill in the common byte stack. This will generate two identical GLOBCNT values when decoded.

Current Encoding Buffer
05 00 00 00 00 00 52 05 06 **01 10**

This results in encodings in the encoding buffer to generate all GLOBCNT values in the **GLOBSET**. To complete the encoding, an **End** command has to be added. Before the **End** command can be added, any bytes on the common byte stack have to be removed. Because all bytes on the common byte stack were pushed with a single **Push** command, only one **Pop** command is needed to remove them.

Current Encoding Buffer
05 00 00 00 00 00 52 05 06 01 10 **50**

The **End** command can now be added.

Current Encoding Buffer
05 00 00 00 00 00 52 05 06 01 10 50 **00**

The GLOBSET 1 encoding can be added to the serialization buffer to produce the following:

Serialization Buffer
01 00 **05 00 00 00 00 00 52 05 06 01 10 50 00** 02 00 <encoded GLOBSET 2>

The last step is to encode GLOBSET 2. GLOBSET 2 contains two GLOBCNT values. The following table is a list of all the GLOBCNT values in the order in which they have to be encoded.

#	GLOBCNT
1	00 00 00 00 00 09
2	00 00 00 00 00 09

Because both GLOBCNT values 1 and 2 are identical, the **Push** command can be used, followed by the full 6 bytes to add to the common byte stack. Because this will fill the common array, it will generate two identical GLOBCNT values when decoded, producing a singleton GLOBCNT range.

Current Encoding Buffer
06 00 00 00 00 00 09

Encodings in the encoding buffer now exist to generate all GLOBCNT values in the **GLOBSET**. To complete the encoding, an **End** command has to be added.

Current Encoding Buffer
06 00 00 00 00 00 09 **00**

The GLOBSET 2 encoding can be added to the serialization buffer to produce the following:

Serialization Buffer
01 00 05 00 00 00 00 00 52 05 06 01 10 50 00 02 00 **06 00 00 00 00 00 09 00**

This completes the serialization of the **IDSET**.

4.5 FastTransfer Stream Produced by a Content Synchronization Download

The following example shows the sample output of a FastTransfer stream that is downloaded to a client during a content synchronization operation. The download operation was configured by using the **RopSynchronizationConfigure** ROP (section 2.2.3.2.1.1) command with the following fields specified in the request buffer:

Field of the request buffer	Value
SynchronizationType	Contents

Field of the request buffer	Value
SendOptions	Unicode, RecoverMode, ForceUnicode, PartialItem<30>
SynchronizationFlag	Unicode, ReadState, FAI, Normal, NoForeignIdentifiers, BestBody, Progress
RestrictionDataSize	0
RestrictionData	< missing >
SynchronizationExtraFlag	Eid, CN, OrderByDeliveryTime

The FastTransfer stream contains the full message change for one message, message deletions, message read state changes, and the final ICS state. The following list shows the structure of the data included in this FastTransfer stream. The list shows the markers that occur in this stream in the order of their appearance. The nesting structure shows the logical relationship of the data delimited by the markers.

```
PidTagIncrSyncProgressMode
    PidTagIncrSyncProgressPerMsg
    PidTagIncrSyncChg
        PidTagIncrSyncMessage
            PidTagStartRecip
            PidTagEndToRecip
            PidTagNewAttach
                PidTagStartEmbed
                    PidTagStartRecip
                    PidTagEndToRecip
                PidTagEndEmbed
            PidTagEndAttach
    PidTagIncrSyncDel
    PidTagIncrSyncRead
    PidTagIncrSyncStateBegin
    PidTagIncrSyncStateEnd
PidTagIncrSyncEnd
```

In the following table, certain property tags are identified as special property tags, which means that they contain 0000 for a **property ID**, and the meaning of the property is determined by the context of the property in the stream.

Bytes on the wire	Value/description
0B 00 74 40	**marker** **PidTagIncrSyncProgressMode** marker (section 2.2.4.1.4) (4074000B [Bool])
02 01 00 00	propDef ProgressInformation (special) (00000102 [Binary])
20 00 00 00	length 32 (0x20)
26 00 00 00- 32 54 76 98 BE BA BE	varSizeValue

Bytes on the wire	Value/description
BA-BE BA BE BA EF CD AB 00-00 00 00 00 EF CD AB 90-78 56 34 12	
0B 00 75 40	**marker** **PidTagIncrSyncProgressPerMsg** marker (section 2.2.4.1.4) (4075000B [Bool])
03 00 00 00	propDef MessageSize (special) (00000003 [Int32])
38 00 00 00	fixedSizeValue [Int32] 56
0B 00 00 00	propDef IsAssociated (special) (0000000B [Bool])
00 00	fixedSizeValue [Bool] FALSE
03 00 12 40	**marker** **PidTagIncrSyncChg marker (section 2.2.4.1.4) (40120003 [Int32])**
02 01 E0 65	propDef **PidTagSourceKey** property (section 2.2.1.2.5) (65E00102 [Binary])
16 00 00 00	length 22 (0x16)
19 D7 FB 0F-06 16 A1 41 BF F6 91 C7-63 DA A8 66 00 00 00 78-2E 21	varSizeValue ....A ....c..f ...x.!
40 00 08 30	propDef **PidTagLastModificationTime** property ([MS-OXPROPS] section 2.861) (30080040 [SysTime])
FC 65 69 CF-C0 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T04:15:02.8437500
02 01 E2 65	propDef **PidTagChangeKey** property (section 2.2.1.2.7) (65E20102 [Binary])
16 00 00 00	length 22 (0x16)
19 D7 FB 0F-06 16 A1 41	varSizeValue ....A

Bytes on the wire	Value/description
F F6 91 C7-63 DA A8 66 00 00 00 78-4D 1C	....c..f ...xM.
02 01 E3 65	propDef **PidTagPredecessorChangeList** property (section 2.2.1.2.8) (65E30102 [Binary])
17 00 00 00	length 23 (0x17)
16 19 D7 FB-0F 06 16 A1 41 BF F6 91-C7 63 DA A8 66 00 00 00-78 4D 1C	varSizeValue ..... A....c.. f...xM.
0B 00 AA 67	propDef **PidTagAssociated** property (section 2.2.1.4) (67AA000B [Bool])
00 00	fixedSizeValue [Bool] False
14 00 4A 67	propDef **PidTagMid** property (section 2.2.1.2.1) (674A0014 [Int64])
01 00 00 00-00 78 2E 21	fixedSizeValue [Int64] 2390980393575645185
14 00 A4 67	propDef **PidTagChangeNumber** property (section 2.2.1.2.3) (67A40014 [Int64])
01 00 00 00-00 78 4D 1C	fixedSizeValue [Int64] 2039418147664035841
03 00 15 40	**marker** **PidTagIncrSyncMessage** marker (section 2.2.4.1.4) (40150003 [Int32])
0B 00 02 00	propDef **PidTagAlternateRecipientAllowed** property ([MS-OXPROPS] section 2.641) (0002000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 17 00	propDef **PidTagImportance** property ([MS-OXCMSG] section 2.2.1.11) (00170003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
1F 00 1A 00	propDef

Bytes on the wire	Value/description
	PidTagMessageClass property ([MS-OXCMSG] section 2.2.1.3) (001A001F [Unicode])
12 00 00 00	length 18 (0x12)
49 00 50 00-4D 00 2E 00 4E 00 6F 00-74 00 65 00 00 00	varSizeValue I.P.M... N.o.t.e. ..
0B 00 23 00	propDef **PidTagOriginatorDeliveryReportRequested** property ([MS-OXOMSG] section 2.2.1.20) (0023000B [Bool])
00 00	fixedSizeValue [Bool] False
03 00 26 00	propDef **PidTagPriority** property ([MS-OXCMSG] section 2.2.1.12) (00260003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 29 00	propDef **PidTagReadReceiptRequested** property ([MS-OXOMSG] section 2.2.1.28) (0029000B [Bool])
00 00	fixedSizeValue [Bool] False
03 00 36 00	propDef **PidTagSensitivity** property ([MS-OXCMSG] section 2.2.1.13) (00360003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
1F 00 37 00	propDef **PidTagSubject** property ([MS-OXPROPS] section 2.1145) (0037001F [Unicode])
26 00 00 00	length 38 (0x26)
54 00 65 00-73 00 74 00 20 00 77 00-69 00 74 00 68 00 20 00-65 00 6D 00 62 00 65 00-64 00 64 00 65 00 64 00-00 00	varSizeValue T.e.s.t. .w.i.t. h..e.m. b.e.d.d. e.d...

Bytes on the wire	Value/description
... value truncated...	
40 00 39 00	propDef **PidTagClientSubmitTime** property ([MS-OXOMSG] section 2.2.3.11) (00390040 [SysTime])
80 BA A7 B7-BC 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T03:45:45.0000000
02 01 3B 00	propDef **PidTagSentRepresentingSearchKey** property ([MS-OXOMSG] section 2.2.1.49) (003B0102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F-4F 3D 46 49 52 53 54 20-4F 52 47 41 4E 49 5A 41-54 49 4F 4E 2F 4F 55 3D-45 58 43 48 41 4E 47 45-20 41 44 4D	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM
... value truncated...	
1F 00 3D 00	propDef **PidTagSubjectPrefix** property ([MS-OXCMSG] section 2.2.1.9) (003D001F [Unicode])
02 00 00 00	length 2 (0x2)
00 00	varSizeValue ..
02 01 3F 00	propDef **PidTagReceivedByEntryId** property ([MS-OXOMSG] section 2.2.1.31) (003F0102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00-DC A7 40 C8 C0 42 10 1A-B4 B9 08 00 2B 2F E1 82-01 00 00 00 00 00 00 00-2F 4F 3D 46	varSizeValue ......@. .B...... +/...... ..../O=F IRST ORG

Bytes on the wire	Value/description
49 52 53 54- 20 4F 52 47	
... value truncated...	
1F 00 40 00	propDef **PidTagReceivedByName** property ([MS-OXOMSG] section 2.2.1.32) (0040001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00- 00 00	varSizeValue t.1...
02 01 41 00	propDef **PidTagSentRepresentingEntryId** property ([MS-OXOMSG] section 2.2.1.47) (00410102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00- DC A7 40 C8 C0 42 10 1A- B4 B9 08 00 2B 2F E1 82- 01 00 00 00 00 00 00 00- 2F 4F 3D 46 49 52 53 54- 20 4F 52 47	varSizeValue .....@. .B..... +/..... ..../O=F IRST ORG
... value truncated...	
1F 00 42 00	propDef **PidTagSentRepresentingName** property ([MS-OXOMSG] section 2.2.1.48) (0042001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00- 00 00	varSizeValue t.1...
02 01 43 00	propDef **PidTagReceivedRepresentingEntryId** property ([MS-OXOMSG] section 2.2.1.47) (00430102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00- DC A7 40 C8	varSizeValue .....@.

Bytes on the wire	Value/description
C0 42 10 1A- B4 B9 08 00 2B 2F E1 82- 01 00 00 00 00 00 00 00- 2F 4F 3D 46 49 52 53 54- 20 4F 52 47	.B..... +/..... ..../O=F IRST ORG
... value truncated...	
1F 00 44 00	propDef **PidTagReceivedRepresentingName** property ([MS-OXOMSG] section 2.2.1.48) (0044001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00- 00 00	varSizeValue t.1...
02 01 51 00	propDef **PidTagReceivedBySearchKey** property ([MS-OXOMSG] section 2.2.1.33) (00510102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F- 4F 3D 46 49 52 53 54 20- 4F 52 47 41 4E 49 5A 41- 54 49 4F 4E 2F 4F 55 3D- 45 58 43 48 41 4E 47 45- 20 41 44 4D	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM
... value truncated...	
02 01 52 00	propDef **PidTagReceivedRepresentingSearchKey** property ([MS-OXOMSG] section 2.2.1.27) (00520102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F- 4F 3D 46 49 52 53 54 20- 4F 52 47 41 4E 49 5A 41-	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH

Bytes on the wire	Value/description
54 49 4F 4E 2F 4F 55 3D-45 58 43 48 41 4E 47 45-20 41 44 4D	ANGE ADM
... value truncated...	
1F 00 64 00	propDef **PidTagSentRepresentingAddressType** property ([MS-OXOMSG] section 2.2.1.45) (0064001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 65 00	propDef **PidTagSentRepresentingEmailAddress** ([MS-OXOMSG] section 2.2.1.46) (0065001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 70 00	propDef **PidTagConversationTopic** property ([MS-OXOMSG] section 2.2.1.5) (0070001F [Unicode])
26 00 00 00	length 38 (0x26)
54 00 65 00-73 00 74 00 20 00 77 00-69 00 74 00 68 00 20 00-65 00 6D 00 62 00 65 00-64 00 64 00	varSizeValue T.e.s.t. .w.i.t. h..e.m. b.e.d.d. e.d...

Bytes on the wire	Value/description
65 00 64 00-00 00	
... value truncated...	
02 01 71 00	propDef **PidTagConversationIndex** property ([MS-OXOMSG] section 2.2.1.3) (00710102 [Binary])
16 00 00 00	length 22 (0x16)
01 C8 84 BC-B6 CB 8A CC 1E B8 32 77-43 2B A1 C6 83 9A 4A F4-BC 14	varSizeValue ....2wC+.. ..J...
1F 00 75 00	propDef **PidTagReceivedByAddressType** ([MS-OXOMSG] section 2.2.1.29) (0075001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 76 00	propDef **PidTagReceivedByEmailAddress** property ([MS-OXOMSG] section 2.2.1.30) (0076001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 77 00	propDef **PidTagReceivedRepresentingAddressType** property ([MS-OXOMSG] section 2.2.1.23) (0077001F [Unicode])
06 00 00 00	length

Bytes on the wire	Value/description
	6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 78 00	propDef **PidTagReceivedRepresentingEmailAddress** property ([MS-OXOMSG] section 2.2.1.24) (0078001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 7D 00	propDef **PidTagTransportMessageHeaders** property ([MS-OXOMSG] section 2.2.1.51) (007D001F [Unicode])
E8 06 00 00	length 1768 (0x6E8)
52 00 65 00-63 00 65 00 69 00 76 00-65 00 64 00 3A 00 20 00-66 00 72 00 6F 00 6D 00-20 00 45 00 58 00 43 00-48 00 2D 00	varSizeValue R.e.c.e. i.v.e.d. :..f.r. o.m..E. X.C.H.-.
... value truncated...	
02 01 7F 00	propDef **PidTagTnefCorrelationKey** property ([MS-OXPROPS] section 2.1160) (007F0102 [Binary])
56 00 00 00	length 86 (0x56)
3C 31 39 44-37 46 42 30	varSizeValue

Bytes on the wire	Value/description
46 30 36 31- 36 41 31 34 31 42 46 46- 36 39 31 43 37 36 33 44- 41 41 38 36 36 37 38 34- 34 42 37 40	<19D7FB0 F0616A14 1BFF691C 763DAA86 67844B7@
... value truncated...	
02 01 19 0C	propDef **PidTagSenderEntryId** property ([MS-OXOMSG] section 2.2.1.42) (0C190102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00- DC A7 40 C8 C0 42 10 1A- B4 B9 08 00 2B 2F E1 82- 01 00 00 00 00 00 00 00- 2F 4F 3D 46 49 52 53 54- 20 4F 52 47	varSizeValue .....@. .B..... +/..... ..../O=F IRST ORG
... value truncated...	
1F 00 1A 0C	propDef **PidTagSenderName** property ([MS-OXOMSG] section 2.2.1.43) (0C1A001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00- 00 00	varSizeValue t.1...
02 01 1D 0C	propDef **PidTagSenderSearchKey** property ([MS-OXOMSG] section 2.2.1.44) (0C1D0102 [Binary])
60 00 00 00	Length 96 (0x60)
45 58 3A 2F- 4F 3D 46 49 52 53 54 20- 4F 52 47 41 4E 49 5A 41- 54 49 4F 4E 2F 4F 55 3D-	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM

Bytes on the wire	Value/description
45 58 43 48 41 4E 47 45- 20 41 44 4D	
... value truncated...	
1F 00 1E 0C	propDef **PidTagSenderAddressType** property ([MS-OXOMSG] section 2.2.1.40) (0C1E001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00- 00 00	varSizeValue E.X...
1F 00 1F 0C	propDef **PidTagSenderEmailAddress** property ([MS-OXOMSG] section 2.2.1.41) (0C1F001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00- 3D 00 46 00 49 00 52 00- 53 00 54 00 20 00 4F 00- 52 00 47 00 41 00 4E 00- 49 00 5A 00 41 00 54 00- 49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
03 00 D3 83- 03 20 06 00 00 00 00 00- C0 00 00 00 00 00 00 46- 00 2A 81 00 00	propDef **PidLidTaskAcceptanceState** property ([MS-OXOTASK] section 2.2.2.2.30) (0x812A [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 D2 83- 03 20 06 00 00 00 00 00- C0 00 00 00 00 00 00 46- 00 2C 81 00	propDef **PidLidTaskFFixOffline** property ([MS-OXOTASK] section 2.2.2.2.31) (0x812C [PSETID_Task]) [Bool]

Bytes on the wire	Value/description
00	
00 00	fixedSizeValue [Bool] False
0B 00 D1 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 24 81 00 00	propDef **PidLidTaskNoCompute** property ([MS-OXOTASK] section 2.2.2.2.35) (0x8124 [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] False
40 00 06 0E	propDef **PidTagMessageDeliveryTime** property ([MS-OXOMSG] section 2.2.3.9) (0E060040 [SysTime])
80 E7 D8 B8-BC 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T03:45:47.0000000
03 00 07 0E	propDef **PidTagMessageFlags** property ([MS-OXCMSG] section 2.2.1.6) (0E070003 [Int32])
31 00 00 00	fixedSizeValue [Int32] 49
03 00 CE 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 29 81 00 00	propDef **PidLidTaskOwnership** property ([MS-OXOTASK] section 2.2.2.2.29) (0x8129 [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
03 00 17 0E	propDef **PidTagMessageStatus** property ([MS-OXCMSG] section 2.2.1.8) (0E170003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 D0 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 11 81 00 00	propDef **PidLidTaskEstimatedEffort** property ([MS-OXOTASK] section 2.2.2.2.12) (0x8111 [PSETID_Task]) [Int32]

Bytes on the wire	Value/description
00 00 00 00	fixedSizeValue [Int32] 0
1F 00 1D 0E	propDef **PidTagNormalizedSubject** property ([MS-OXCMSG] section 2.2.1.10) (0E1D001F [Unicode])
26 00 00 00	length 38 (0x26)
54 00 65 00-73 00 74 00 20 00 77 00-69 00 74 00 68 00 20 00-65 00 6D 00 62 00 65 00-64 00 64 00 65 00 64 00-00 00	varSizeValue T.e.s.t. .w.i.t. h..e.m. b.e.d.d. e.d...
... value truncated...	
0B 00 1F 0E	propDef **PidTagRtfInSync** property ([MS-OXCMSG] section 2.2.1.44.5) (0E1F000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 23 0E	propDef **PidTagInternetArticleNumber** property ([MS-OXPROPS] section 2.842) (0E230003 [Int32])
26 00 00 00	fixedSizeValue [Int32] 38
03 00 79 0E	propDef **PidTagTrustSender** property ([MS-OXPROPS] section 2.1164) (0E790003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
03 00 CF 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 10 81 00 00	propDef **PidLidTaskActualEffort** property ([MS-OXOTASK] section 2.2.2.2.11) (0x8110 [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0

Bytes on the wire	Value/description
03 00 F7 0F	propDef **PidTagAccessLevel** property ([MS-OXCPRPT] section 2.2.1.2) (0FF70003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
1F 00 CD 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 21 81 00 00	propDef **PidLidTaskAssigner** property ([MS-OXOTASK] section 2.2.2.2.24) (0x8121 [PSETID_Task]) [Unicode]
02 00 00 00	length 2 (0x2)
00 00	varSizeValue ..
03 00 CC 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 23 81 00 00	propDef **PidLidTaskOrdinal** property ([MS-OXOTASK] section 2.2.2.2.26) (0x8123 [PSETID_Task]) [Int32]
FF FF FF 7F	fixedSizeValue [Int32] 2147483647
1F 00 35 10	propDef **PidTagInternetMessageId** property ([MS-OXOMSG] section 2.2.1.12) (1035001F [Unicode])
AC 00 00 00	length 172 (0xAC)
3C 00 31 00-39 00 44 00 37 00 46 00-42 00 30 00 46 00 30 00-36 00 31 00 36 00 41 00-31 00 34 00 31 00 42 00-46 00 46 00	varSizeValue <.1.9.D. 7.F.B.0. F.0.6.1. 6.A.1.4. 1.B.F.F.
... value truncated...	
03 00 80 10	propDef

Bytes on the wire	Value/description
	PidTagIconIndex property ([MS-OXOMSG] section 2.2.1.10) (10800003 [Int32])
FF FF FF FF	fixedSizeValue [Int32] -1
40 00 07 30	propDef **PidTagCreationTime** property ([MS-OXCMSG] section 2.2.2.3) (30070040 [SysTime])
A2 DA EF B9-BC 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T03:45:48.8281250
40 00 08 30	propDef **PidTagLastModificationTime** property (30080040 [SysTime])
FC 65 69 CF-C0 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T04:15:02.8437500
02 01 0B 30	propDef **PidTagSearchKey** property ([MS-OXCPRPT] section 2.2.1.9) (300B0102 [Binary])
10 00 00 00	length 16 (0x10)
6B 3B AA B8-C7 83 78 4E 80 8E F2 DE-04 82 C8 EB	varSizeValue k;....xN .....
0B 00 40 3A	propDef **PidTagSendRichInfo** property ([MS-OXOMSG] section 2.2.1.39) (3A40000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 DE 3F	propDef **PidTagInternetCodepage** property ([MS-OXCMSG] section 2.2.1.44.6) (3FDE0003 [Int32])
9F 4E 00 00	fixedSizeValue [Int32] 20127
03 00 F1 3F	propDef **PidTagMessageLocaleId** property ([MS-OXCMSG] section 2.2.1.5) (3FF10003 [Int32])
09 04 00 00	fixedSizeValue [Int32] 1033
03 00 FD 3F	propDef **PidTagMessageCodepage** property ([MS-OXCMSG] section 2.2.1.4) (3FFD0003 [Int32])
E3 04 00 00	fixedSizeValue [Int32] 1251

Bytes on the wire	Value/description
03 00 19 40	propDef **PidTagSenderFlags** property ([MS-OXPROPS] section 2.1106) (40190003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 1A 40	propDef **PidTagSentRepresentingFlags** property ([MS-OXPROPS] section 2.1120) (401A0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 1B 40	propDef **PidTagReceivedByFlags** property ([MS-OXPROPS] section 2.985) (401B0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 1C 40	propDef **PidTagReceivedRepresentingFlags** property ([MS-OXPROPS] section 2.991) (401C0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 76 40	propDef **PidTagContentFilterSpamConfidenceLevel** property ([MS-OXCSPAM] section 2.2.1.3) (40760003 [Int32])
FF FF FF FF	fixedSizeValue [Int32] -1
03 00 02 59	propDef **PidTagInternetMailOverrideFormat** property ([MS-OXOMSG] section 2.2.1.11) (59020003 [Int32])
00 00 16 00	fixedSizeValue [Int32] 1441792
03 00 09 59	propDef **PidTagMessageEditorFormat** property ([MS-OXPROPS] section 2.887) (59090003 [Int32])
02 00 00 00	fixedSizeValue [Int32] 2
03 00 C6 65	propDef **PidTagSecureSubmitFlags** property ([MS-OXPROPS] section 2.1099) (65C60003 [Int32])
02 00 00 00	fixedSizeValue [Int32] 2

Bytes on the wire	Value/description
1F 00 D4 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 27 81 00 00	propDef **PidLidTaskRole** property ([MS-OXPROPS] section 2.330) (0x8127 [PSETID_Task]) [Unicode]
02 00 00 00	length 2 (0x2)
00 00	varSizeValue ..
0B 00 D5 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 03 81 00 00	propDef **PidLidTeamTask** property ([MS-OXOTASK] section 2.2.2.2.36) (0x8103 [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
0B 00 D6 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 26 81 00 00	propDef **PidLidTaskFRecurring** property ([MS-OXOTASK] section 2.2.2.2.28) (0x8126 [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
03 00 00 80-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 52 85 00 00	propDef **PidLidCurrentVersion** property ([MS-OXPROPS] section 2.83) (0x8552 [PSETID_Common]) [Int32]
04 ED 01 00	fixedSizeValue [Int32] 126212
1F 00 01 80-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-	propDef **PidLidCurrentVersionName** property ([MS-OXPROPS] section 2.84) (0x8554 [PSETID_Common]) [Unicode]

Bytes on the wire	Value/description
00 54 85 00 00	
0A 00 00 00	length 10 (0xA)
31 00 32 00- 2E 00 30 00 00 00	varSizeValue 1.2...0. ..
03 00 02 80- 08 20 06 00 00 00 00 00- C0 00 00 00 00 00 00 46- 00 10 85 00 00	propDef **PidLidSideEffects** property ([MS-OXCMSG] section 2.2.1.16) (0x8510 [PSETID_Common]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 08 80- 08 20 06 00 00 00 00 00- C0 00 00 00 00 00 00 46- 00 03 85 00 00	propDef **PidLidReminderSet** property ([MS-OXORMDR] section 2.2.1.1) (0x8503 [PSETID_Common]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
1F 10 0C 80- 29 03 02 00 00 00 00 00- C0 00 00 00 00 00 00 46- 01 4B 00 65 00 79 00 77- 00 6F 00 72 00 64 00 73- 00 00 00	propDef **PidNameKeywords** property ([MS-OXCMSG] section 2.2.1.17) (Keywords [PS_PUBLIC_STRINGS]) [MultiValueUnicode]
02 00 00 00	length 2 (0x2)
1C 00 00 00	length 28 (0x1C)
42 00 6C 00- 75 00 65 00 20 00 43 00-	varSizeValue B.l.u.e. .C.a.t.

Bytes on the wire	Value/description
61 00 74 00 65 00 67 00- 6F 00 72 00 79 00 00 00	e.g.o.r. y...
20 00 00 00	length 32 (0x20)
59 00 65 00- 6C 00 6C 00 6F 00 77 00- 20 00 43 00 61 00 74 00- 65 00 67 00 6F 00 72 00- 79 00 00 00	varSizeValue Y.e.l.l. o.w..C. a.t.e.g. o.r.y...
0B 00 4D 81- 08 20 06 00 00 00 00 00- C0 00 00 00 00 00 00 46- 00 0E 85 00 00	propDef **PidLidAgingDontAgeMe** property ([MS-OXPROPS] section 2.4) (0x850E [PSETID_Common]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
03 00 84 81- 08 20 06 00 00 00 00 00- C0 00 00 00 00 00 00 46- 00 18 85 00 00	propDef **PidLidTaskMode** property ([MS-OXOTASK] section 2.2.2.2.1) (0x8518 [PSETID_Common]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 4B 82- 08 20 06 00 00 00 00 00- C0 00 00 00 00 00 00 46- 00 06 85 00 00	propDef **PidLidPrivate** property ([MS-OXCMSG] section 2.2.1.15) (0x8506 [PSETID_Common]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
1F 00 4D 82- 08 20 06 00	propDef **PidLidInternetAccountName** property ([MS-OXOMSG] section 2.2.1.52) (0x8580

Bytes on the wire	Value/description
00 00 00 00-C0 00 00 00 00 00 00 46-00 80 85 00 00	[PSETID_Common]) [Unicode]
26 00 00 00	length 38 (0x26)
4D 00 69 00-63 00 72 00 6F 00 73 00-6F 00 66 00 74 00 20 00-45 00 78 00 63 00 68 00-61 00 6E 00 67 00 65 00-00 00	varSizeValue M.i.c.r. o.s.o.f. t..E.x. c.h.a.n. g.e...
... value truncated...	
1F 00 4E 82-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 81 85 00 00	propDef **PidLidInternetAccountStamp** property ([MS-OXOMSG] section 2.2.1.53) (0x8581 [PSETID_Common]) [Unicode]
E4 00 00 00	length 228 (0xE4)
30 00 30 00-30 00 30 00 30 00 30 00-30 00 32 00 01 00 45 00-58 00 43 00 48 00 2D 00-43 00 4C 00 49 00 2D 00-31 00 38 00	varSizeValue 0.0.0.0. 0.0.0.2. ..E.X.C. H.-.C.L. I.-.1.8.
... value truncated...	
0B 00 4F 82-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 82 85 00	propDef **PidLidUseTnef** property ([MS-OXOMSG] section 2.2.1.56) (0x8582 [PSETID_Common]) [Bool]

Bytes on the wire	Value/description
00	
00 00	fixedSizeValue [Bool] FALSE
03 00 A8 83-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 01 85 00 00	propDef **PidLidReminderDelta** property ([MS-OXORMDR] section 2.2.1.3) (0x8501 [PSETID_Common]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
03 00 AD 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 01 81 00 00	propDef **PidLidTaskStatus** property ([MS-OXOTASK] section 2.2.2.2.2) (0x8101 [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
05 00 AE 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 02 81 00 00	propDef **PidLidPercentComplete** property ([MS-OXOTASK] section 2.2.2.2.3) (0x8102 [PSETID_Task]) [Double]
00 00 00 00-00 00 00 00	fixedSizeValue [Double] 0
0B 00 B0 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 1C 81 00 00	propDef **PidLidTaskComplete** property ([MS-OXOTASK] section 2.2.2.2.20) (0x811C [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
03 00 CA 83-03 20 06 00 00 00 00 00-C0 00 00 00	propDef **PidLidTaskState** property ([MS-OXOTASK] section 2.2.2.2.14) (0x8113 [PSETID_Task]) [Int32]

Bytes on the wire	Value/description
00 00 00 46-00 13 81 00 00	
01 00 00 00	fixedSizeValue [Int32] 1
03 00 CB 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 12 81 00 00	propDef **PidLidTaskVersion** property ([MS-OXOTASK] section 2.2.2.2.13) (0x8112 [PSETID_Task]) [Int32]
01 00 00 00	fixedSizeValue [Int32] 1
02 01 13 10	propDef **PidTagBodyHtml** property ([MS-OXCMSG] section 2.2.1.44.3) (10130102 [Binary])
58 06 00 00	length 1624 (0x658)
3C 68 74 6D-6C 20 78 6D 6C 6E 73 3A-76 3D 22 75 72 6E 3A 73-63 68 65 6D 61 73 2D 6D-69 63 72 6F 73 6F 66 74-2D 63 6F 6D	varSizeValue <html xm lns:v="u rn:schem as-micro soft-com
... value truncated...	
03 00 16 40	propDef **PidTagFXDelProp** property (section 2.2.4.1.5.1) (40160003 [Int32])
0D 00 12 0E	fixedSizeValue **PidTagMessageRecipients** property ([MS-OXPROPS] section 2.892) (0E12000D [Object])
03 00 03 40	**marker** **PidTagStartRecip** marker (section 2.2.4.1.4) (40030003 [Int32])
03 00 00 30	propDef **PidTagRowid** property ([MS-OXPROPS] section 2.1035) (30000003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0

Bytes on the wire	Value/description
1F 00 02 30	propDef **PidTagAddressType** property ([MS-OXCMAIL] section 2.1.3.1.9) (3002001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 03 30	propDef **PidTagEmailAddress** property ([MS-OXPROPS] section 2.752) (3003001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 01 30	propDef **PidTagDisplayName** property ([MS-OXCFOLD] section 2.3.2.2.3) (3001001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
02 01 F6 0F	propDef **PidTagInstanceKey** property ([MS-OXPROPS] section 2.839) (0FF60102 [Binary])
04 00 00 00	length 4 (0x4)
00 00 00 00	varSizeValue ....
03 00 15 0C	propDef **PidTagRecipientType** property ([MS-OXOMSG] section 2.2.3.1) (0C150003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
02 01 FF 0F	propDef

Bytes on the wire	Value/description
	PidTagEntryId property ([MS-OXPROPS] section 2.759) (0FFF0102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00-DC A7 40 C8 C0 42 10 1A-B4 B9 08 00 2B 2F E1 82-01 00 00 00 00 00 00 00-2F 4F 3D 46 49 52 53 54-20 4F 52 47	varSizeValue .....@. .B..... +/..... ..../O=F IRST ORG
... value truncated...	
02 01 0B 30	propDef **PidTagSearchKey** property (300B0102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F-4F 3D 46 49 52 53 54 20-4F 52 47 41 4E 49 5A 41-54 49 4F 4E 2F 4F 55 3D-45 58 43 48	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH
... value truncated...	
1F 00 20 3A	propDef **PidTagTransmittableDisplayName** property ([MS-OXOABK] section 2.2.3.8) (3A20001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
0B 00 0F 0E	propDef **PidTagResponsibility** property ([MS-OXPROPS] section 2.1027) (0E0F000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
0B 00 40 3A	propDef **PidTagSendRichInfo** property (3A40000B [Bool])

Bytes on the wire	Value/description
01 00	fixedSizeValue [Bool] TRUE
03 00 FD 5F	propDef **PidTagRecipientFlags** property ([MS-OXOCAL] section 2.2.4.9.1) (5FFD0003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
02 01 F7 5F	propDef **PidTagRecipientEntryId** property ([MS-OXPROPS] section 2.996) (5FF70102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00-DC A7 40 C8 C0 42 10 1A-B4 B9 08 00 2B 2F E1 82-01 00 00 00 00 00 00 00-2F 6F 3D 46 69 72 73 74-20 4F 72 67	varSizeValue ......@. .B...... +/..... ..../o=F irst Org
... value truncated...	
1F 00 FE 39	propDef **PidTagPrimarySmtpAddress** property ([MS-OXPROPS] section 2.963) (39FE001F [Unicode])
46 00 00 00	length 70 (0x46)
74 00 31 00-40 00 65 00 75 00 6D 00-61 00 72 00 75 00 2D 00-64 00 6F 00 6D 00 2E 00-65 00 78 00 74 00 65 00-73 00 74 00	varSizeValue t.1.@.e. u.m.a.r. u.-.d.o. m...e.x. t.e.s.t.
... value truncated...	
03 00 05 39	propDef **PidTagDisplayTypeEx** property ([MS-OXOABK] section 2.2.3.12) (39050003 [Int32])
00 00 00 40	fixedSizeValue [Int32] 1073741824

Bytes on the wire	Value/description
03 00 00 39	propDef **PidTagDisplayType** property ([MS-OXOABK] section 2.2.3.11) (39000003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 FE 0F	propDef **PidTagObjectType** property ([MS-OXOABK] section 2.2.3.10) (0FFE0003 [Int32])
06 00 00 00	fixedSizeValue [Int32] 6
1F 00 FF 39	propDef **PidTag7BitDisplayName** property ([MS-OXOABK] section 2.2.3.7) (39FF001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
1F 00 00 3A	propDef **PidTagAccount** property ([MS-OXOABK] section 2.2.3.20) (3A00001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
03 00 FF 5F	propDef **PidTagRecipientTrackStatus** property ([MS-OXOCAL] section 2.2.4.9.2) (5FFF0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 DE 5F	propDef **PidTagRecipientResourceState** property ([MS-OXPROPS] section 2.1003) (5FDE0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
1F 00 F6 5F	propDef **PidTagRecipientDisplayName** property ([MS-OXPROPS] section 2.995) (5FF6001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-	varSizeValue

Bytes on the wire	Value/description
00 00	t.1...
03 00 DF 5F	propDef **PidTagRecipientOrder** property ([MS-OXPROPS] section 2.998) (5FDF0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 04 40	**marker** **PidTagEndToRecip marker (section 2.2.4.1.4) (40040003 [Int32])**
03 00 16 40	propDef **PidTagFXDelProp** property (40160003 [Int32])
0D 00 13 0E	FixedSizeValue **PidTagMessageAttachments** property ([MS-OXPROPS] section 2.882) (0E13000D [Object])
03 00 00 40	**marker** **PidTagNewAttach marker (section 2.2.4.1.4) (40000003 [Int32])**
03 00 21 0E	propDef **PidTagAttachNumber** property ([MS-OXCMSG] section 2.2.2.6) (0E210003 [Int32])
00 00 00 00	marker [Int32] 0
02 01 02 37	propDef **PidTagAttachEncoding** property ([MS-OXCMSG] section 2.2.2.20) (37020102 [Binary])
00 00 00 00	length 0 (0x0)
03 00 0B 37	propDef **PidTagRenderingPosition** property ([MS-OXCMSG] section 2.2.2.16) (370B0003 [Int32])
FF FF FF FF	fixedSizeValue [Int32] -1
03 00 20 0E	propDef **PidTagAttachSize** property ([MS-OXCMSG] section 2.2.2.5) (0E200003 [Int32])
E7 15 00 00	fixedSizeValue [Int32] 5607
03 00 F7 0F	propDef **PidTagAccessLevel** property (0FF70003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
40 00 07 30	propDef

Bytes on the wire	Value/description
	PidTagCreationTime property (30070040 [SysTime])
E2 EA E3 B1-BC 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T03:45:35.3281250
40 00 08 30	propDef **PidTagLastModificationTime** property (30080040 [SysTime])
E2 EA E3 B1-BC 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T03:45:35.3281250
03 00 05 37	propDef **PidTagAttachMethod** property ([MS-OXCMSG] section 2.2.2.9) (37050003 [Int32])
05 00 00 00	fixedSizeValue [Int32] 5
02 01 09 37	propDef **PidTagAttachRendering** property ([MS-OXCMSG] section 2.2.2.17) (37090102 [Binary])
B8 0D 00 00	length 3512 (0xDB8)
01 00 09 00-00 03 DC 06 00 00 00 00-21 06 00 00 00 00 05 00-00 00 09 02 00 00 00 00-05 00 00 00 01 02 FF FF-FF 00 A5 00	varSizeValue ....!... ....
... value truncated...	
03 00 14 37	propDef **PidTagAttachFlags** property ([MS-OXCMSG] section 2.2.2.18) (37140003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 FE 7F	propDef **PidTagAttachmentHidden** property ([MS-OXCMSG] section 2.2.2.24) (7FFE000B [Bool])
00 00	fixedSizeValue [Bool] FALSE
1F 00 04 37	propDef **PidTagAttachFilename** property ([MS-OXCMSG] section 2.2.2.11) (3704001F [Unicode])

Bytes on the wire	Value/description
0E 00 00 00	length 14 (0xE)
54 00 65 00- 73 00 74 00 20 00 31 00- 00 00	varSizeValue T.e.s.t. .1...
0B 00 FF 7F	propDef **PidTagAttachmentContactPhoto** property ([MS-OXPROPS] section 2.661) (7FFF000B [Bool])
00 00	fixedSizeValue [Bool] FALSE
1F 00 01 30	propDef **PidTagDisplayName** property (3001001F [Unicode])
0E 00 00 00	length 14 (0xE)
54 00 65 00- 73 00 74 00 20 00 31 00- 00 00	varSizeValue T.e.s.t. .1...
02 01 F9 0F	propDef **PidTagRecordKey** property ([MS-OXPROPS] section 2.1007) (0FF90102 [Binary])
04 00 00 00	length 4 (0x4)
00 00 00 00	varSizeValue ....
03 00 01 40	**marker** **PidTagStartEmbed** marker (section 2.2.4.1.4) (40010003 [Int32])
14 00 4A 67	propDef **PidTagMid** property (section 2.2.1.2.1) (674A0014 [Int64])
01 00 00 00- 00 78 48 C1	fixedSizeValue [Int64] -4519230284670959615
0B 00 02 00	propDef **PidTagAlternateRecipientAllowed** property (0002000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 17 00	propDef **PidTagImportance** property (00170003 [Int32])

Bytes on the wire	Value/description
01 00 00 00	fixedSizeValue [Int32] 1
1F 00 1A 00	propDef **PidTagMessageClass** property (001A001F [Unicode])
12 00 00 00	length 18 (0x12)
49 00 50 00-4D 00 2E 00 4E 00 6F 00-74 00 65 00 00 00	varSizeValue I.P.M... N.o.t.e. ..
0B 00 23 00	propDef **PidTagOriginatorDeliveryReportRequested** property (0023000B [Bool])
00 00	fixedSizeValue [Bool] FALSE
03 00 26 00	propDef **PidTagPriority** property (00260003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 29 00	propDef **PidTagReadReceiptRequested** property (0029000B [Bool])
00 00	fixedSizeValue [Bool] FALSE
03 00 36 00	propDef **PidTagSensitivity** property (00360003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
1F 00 37 00	propDef **PidTagSubject** property (0037001F [Unicode])
0E 00 00 00	length 14 (0xE)
54 00 65 00-73 00 74 00 20 00 31 00-00 00	varSizeValue T.e.s.t. .1...
40 00 39 00	propDef **PidTagClientSubmitTime** property (00390040 [SysTime])

Bytes on the wire	Value/description
00 B4 A1 9D- 8B 84 C8 01	fixedSizeValue [SysTime] 2008-03-12T21:54:16.0000000
02 01 3B 00	propDef **PidTagSentRepresentingSearchKey** property (003B0102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F- 4F 3D 46 49 52 53 54 20- 4F 52 47 41 4E 49 5A 41- 54 49 4F 4E 2F 4F 55 3D- 45 58 43 48 41 4E 47 45- 20 41 44 4D	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM
... value truncated...	
1F 00 3D 00	propDef **PidTagSubjectPrefix** property (003D001F [Unicode])
02 00 00 00	length 2 (0x2)
00 00	varSizeValue ..
02 01 3F 00	propDef **PidTagReceivedByEntryId** property (003F0102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00- DC A7 40 C8 C0 42 10 1A- B4 B9 08 00 2B 2F E1 82- 01 00 00 00 00 00 00 00- 2F 4F 3D 46 49 52 53 54- 20 4F 52 47	varSizeValue .....@. .B..... +/..... ..../O=F IRST ORG
... value truncated...	
1F 00 40 00	propDef **PidTagReceivedByName** property (0040001F [Unicode])

Bytes on the wire	Value/description
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
02 01 41 00	propDef **PidTagSentRepresentingEntryId** property (00410102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00-DC A7 40 C8 C0 42 10 1A-B4 B9 08 00 2B 2F E1 82-01 00 00 00 00 00 00 00-2F 4F 3D 46 49 52 53 54-20 4F 52 47	varSizeValue ......@. .B...... +/..... ..../O=F IRST ORG
... value truncated...	
1F 00 42 00	propDef **PidTagSentRepresentingName** property (0042001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
02 01 43 00	propDef **PidTagReceivedRepresentingEntryId** property (00430102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00-DC A7 40 C8 C0 42 10 1A-B4 B9 08 00 2B 2F E1 82-01 00 00 00 00 00 00 00-2F 4F 3D 46 49 52 53 54-20 4F 52 47	varSizeValue ......@. .B...... +/..... ..../O=F IRST ORG
... value truncated...	

Bytes on the wire	Value/description
1F 00 44 00	propDef **PidTagReceivedRepresentingName** property (0044001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
02 01 51 00	propDef **PidTagReceivedBySearchKey** property (00510102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F-4F 3D 46 49 52 53 54 20-4F 52 47 41 4E 49 5A 41-54 49 4F 4E 2F 4F 55 3D-45 58 43 48 41 4E 47 45-20 41 44 4D	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM
... value truncated...	
02 01 52 00	propDef **PidTagReceivedRepresentingSearchKey** property (00520102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F-4F 3D 46 49 52 53 54 20-4F 52 47 41 4E 49 5A 41-54 49 4F 4E 2F 4F 55 3D-45 58 43 48 41 4E 47 45-20 41 44 4D	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM
... value truncated...	
0B 00 63 00	propDef **PidTagResponseRequested** property ([MS-OXOMSG] section 2.2.1.38) (0063000B [Bool])
01 00	fixedSizeValue [Bool] TRUE

Bytes on the wire	Value/description
1F 00 64 00	propDef **PidTagSentRepresentingAddressType** property (0064001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 65 00	propDef **PidTagSentRepresentingEmailAddress** property (0065001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 70 00	propDef **PidTagConversationTopic** property (0070001F [Unicode])
0E 00 00 00	length 14 (0xE)
54 00 65 00-73 00 74 00 20 00 31 00-00 00	varSizeValue T.e.s.t. .1...
02 01 71 00	propDef **PidTagConversationIndex** property (00710102 [Binary])
16 00 00 00	length 22 (0x16)
01 C8 84 8B-9D B1 08 58 53 52 00 5B-4A D4 96 BA 3C 88 9D B4-16 AE	varSizeValue ....X SR.[J... <.....
1F 00 75 00	propDef

Bytes on the wire	Value/description
	PidTagReceivedByAddressType property (0075001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 76 00	propDef **PidTagReceivedByEmailAddress** property (0076001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 77 00	propDef **PidTagReceivedRepresentingAddressType** property (0077001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 78 00	propDef **PidTagReceivedRepresentingEmailAddress** property (0078001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.

Bytes on the wire	Value/description
... value truncated...	
1F 00 7D 00	propDef **PidTagTransportMessageHeaders** property (007D001F [Unicode])
B0 06 00 00	length 1712 (0x6B0)
52 00 65 00- 63 00 65 00 69 00 76 00- 65 00 64 00 3A 00 20 00- 66 00 72 00 6F 00 6D 00- 20 00 45 00 58 00 43 00- 48 00 2D 00	varSizeValue R.e.c.e. i.v.e.d. :..f.r. o.m..E. X.C.H.-.
... value truncated...	
0B 00 17 0C	propDef **PidTagReplyRequested** property ([MS-OXOMSG] section 2.2.1.37) (0C17000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
02 01 19 0C	propDef **PidTagSenderEntryId** property (0C190102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00- DC A7 40 C8 C0 42 10 1A- B4 B9 08 00 2B 2F E1 82- 01 00 00 00 00 00 00 00- 2F 4F 3D 46 49 52 53 54- 20 4F 52 47	varSizeValue ......@. .B...... +/...... ..../O=F IRST ORG
... value truncated...	
1F 00 1A 0C	propDef **PidTagSenderName** property (0C1A001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-	varSizeValue

Bytes on the wire	Value/description
00 00	t.1...
02 01 1D 0C	propDef **PidTagSenderSearchKey** property (0C1D0102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F- 4F 3D 46 49 52 53 54 20- 4F 52 47 41 4E 49 5A 41- 54 49 4F 4E 2F 4F 55 3D- 45 58 43 48 41 4E 47 45- 20 41 44 4D	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM
... value truncated...	
1F 00 1E 0C	propDef **PidTagSenderAddressType** property (0C1E001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00- 00 00	varSizeValue E.X...
1F 00 1F 0C	propDef **PidTagSenderEmailAddress** property (0C1F001F [Unicode])
BA 00 00 00	length 186 (0xBA)
2F 00 4F 00- 3D 00 46 00 49 00 52 00- 53 00 54 00 20 00 4F 00- 52 00 47 00 41 00 4E 00- 49 00 5A 00 41 00 54 00- 49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 D4 83- 03 20 06 00 00 00 00 00- C0 00 00 00	propDef **PidLidTaskRole** property (0x8127 [PSETID_Task]) [Unicode]

Bytes on the wire	Value/description
00 00 00 46-00 27 81 00 00	
02 00 00 00	length 2 (0x2)
00 00	varSizeValue ..
03 00 D3 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 2A 81 00 00	propDef **PidLidTaskAcceptanceState** property (0x812A [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 D2 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 2C 81 00 00	propDef **PidLidTaskFFixOffline** property (0x812C [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
40 00 06 0E	propDef **PidTagMessageDeliveryTime** property (0E060040 [SysTime])
00 0E 04 A0-8B 84 C8 01	fixedSizeValue [SysTime] 2008-03-12T21:54:20.0000000
03 00 07 0E	propDef **PidTagMessageFlags** property (0E070003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
03 00 CF 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 10 81 00 00	propDef **PidLidTaskActualEffort** property (0x8110 [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue

Bytes on the wire	Value/description
	[Int32] 0
03 00 17 0E	propDef **PidTagMessageStatus** property (0E170003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 D1 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 24 81 00 00	propDef **PidLidTaskNoCompute** property (0x8124 [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
1F 00 1D 0E	propDef **PidTagNormalizedSubject** property (0E1D001F [Unicode])
0E 00 00 00	length 14 (0xE)
54 00 65 00-73 00 74 00 20 00 31 00-00 00	varSizeValue T.e.s.t. .1...
0B 00 1F 0E	propDef **PidTagRtfInSync** property (0E1F000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 23 0E	propDef **PidTagInternetArticleNumber** property (0E230003 [Int32])
1B 00 00 00	fixedSizeValue [Int32] 27
03 00 2B 0E	propDef **PidTagToDoItemFlags** property ([MS-OXOFLAG] section 2.2.1.6) (0E2B0003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
03 00 79 0E	propDef **PidTagTrustSender** property (0E790003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1

Bytes on the wire	Value/description
03 00 D0 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 11 81 00 00	propDef **PidLidTaskEstimatedEffort** property (0x8111 [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
03 00 F7 0F	propDef **PidTagAccessLevel** property (0FF70003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 D6 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 26 81 00 00	propDef **PidLidTaskFRecurring** property (0x8126 [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
02 01 09 10	propDef **PidTagRtfCompressed** property ([MS-OXCMSG] section 2.2.1.44.4) (10090102 [Binary])
22 05 00 00	length 1314 (0x522)
1E 05 00 00-85 0B 00 00 4C 5A 46 75-31 AE 9B E3 03 00 0A 00-72 63 70 67 31 32 35 83-00 50 03 52 68 74 6D 6C-31 03 31 F8	varSizeValue ..... LZFu1... ....rcpg 125..P.R html1.1.
... value truncated...	
0B 00 D5 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-	propDef **PidLidTeamTask** property (0x8103 [PSETID_Task]) [Bool]

Bytes on the wire	Value/description
00 03 81 00 00	
00 00	fixedSizeValue [Bool] FALSE
1F 00 35 10	propDef **PidTagInternetMessageId** property (1035001F [Unicode])
AC 00 00 00	length 172 (0xAC)
3C 00 31 00-39 00 44 00 37 00 46 00-42 00 30 00 46 00 30 00-36 00 31 00 36 00 41 00-31 00 34 00 31 00 42 00-46 00 46 00	varSizeValue <.1.9.D. 7.F.B.0. F.0.6.1. 6.A.1.4. 1.B.F.F.
... value truncated...	
03 00 80 10	propDef **PidTagIconIndex** property (10800003 [Int32])
FF FF FF FF	fixedSizeValue [Int32] -1
03 00 90 10	propDef **PidTagFlagStatus** property ([MS-OXOFLAG] section 2.2.1.1) (10900003 [Int32])
02 00 00 00	fixedSizeValue [Int32] 2
03 00 95 10	propDef **PidTagFollowupIcon** property ([MS-OXOFLAG] section 2.2.1.2) (10950003 [Int32])
06 00 00 00	fixedSizeValue [Int32] 6
40 00 07 30	propDef **PidTagCreationTime** property (30070040 [SysTime])
90 F8 65 B0-BC 84 C8 01	fixedSizeValue [SysTime] 2008-03-13T03:45:32.8250000
40 00 08 30	propDef **PidTagLastModificationTime** property (30080040 [SysTime])
90 F8 65 B0-	fixedSizeValue

Bytes on the wire	Value/description
BC 84 C8 01	[SysTime] 2008-03-13T03:45:32.8250000
02 01 0B 30	propDef **PidTagSearchKey** property (300B0102 [Binary])
10 00 00 00	length 16 (0x10)
87 56 4A B2-FC C2 77 46 A4 81 15 08-9D 47 46 8C	varSizeValue .VJ...wF .....GF.
02 01 10 30	propDef **PidTagTargetEntryId** property ([MS-OXOMSG] section 2.2.1.66) (30100102 [Binary])
46 00 00 00	length 70 (0x46)
00 00 00 00-FE C7 EE E9 76 05 2D 4F-80 00 61 68 94 97 4B 0A-07 00 19 D7 FB 0F 06 16-A1 41 BF F6 91 C7 63 DA-A8 66 00 00	varSizeValue ..... v.-O..ah ..K..... .....A.. ..c..f..
... value truncated...	
0B 00 40 3A	propDef **PidTagSendRichInfo** property (3A40000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 DE 3F	propDef **PidTagInternetCodepage** property (3FDE0003 [Int32])
9F 4E 00 00	fixedSizeValue [Int32] 20127
03 00 F1 3F	propDef **PidTagMessageLocaleId** property (3FF10003 [Int32])
09 04 00 00	fixedSizeValue [Int32] 1033
1F 00 F8 3F	propDef **PidTagCreatorName** property ([MS-OXPROPS] section 2.727) (3FF8001F [Unicode])

Bytes on the wire	Value/description
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
1F 00 FA 3F	propDef **PidTagLastModifierName** property ([MS-OXCPRPT] section 2.2.1.5) (3FFA001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
03 00 FD 3F	propDef **PidTagMessageCodepage** property (3FFD0003 [Int32])
E3 04 00 00	fixedSizeValue [Int32] 1251
03 00 19 40	propDef **PidTagSenderFlags** property (40190003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 1A 40	propDef **PidTagSentRepresentingFlags** property (401A0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 1B 40	propDef **PidTagReceivedByFlags** property (401B0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 1C 40	propDef **PidTagReceivedRepresentingFlags** property (401C0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 76 40	propDef **PidTagContentFilterSpamConfidenceLevel** property (40760003 [Int32])
FF FF FF FF	fixedSizeValue [Int32] -1
03 00 02 59	propDef

Bytes on the wire	Value/description
	PidTagInternetMailOverrideFormat property (59020003 [Int32])
00 00 16 00	fixedSizeValue [Int32] 1441792
03 00 09 59	propDef **PidTagMessageEditorFormat** property (59090003 [Int32])
02 00 00 00	fixedSizeValue [Int32] 2
0B 00 4A 66	propDef **PidTagHasNamedProperties** property ([MS-OXPROPS] section 2.795) (664A000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 02 80-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 10 85 00 00	propDef **PidLidSideEffects** property (0x8510 [PSETID_Common]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 08 80-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 03 85 00 00	propDef **PidLidReminderSet** property (0x8503 [PSETID_Common]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
1F 00 1A 80-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 A4 85 00 00	propDef **PidLidToDoTitle** property ([MS-OXOFLAG] section 2.2.1.12) (0x85A4 [PSETID_Common]) [Unicode]
0E 00 00 00	length 14 (0xE)
54 00 65 00-73 00 74 00	varSizeValue T.e.s.t.

Bytes on the wire	Value/description
20 00 31 00-00 00	.1...
1F 00 2C 80-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 30 85 00 00	propDef **PidLidFlagRequest** property ([MS-OXOFLAG] section 2.2.1.9) (0x8530 [PSETID_Common]) [Unicode]
14 00 00 00	length 20 (0x14)
46 00 6F 00-6C 00 6C 00 6F 00 77 00-20 00 75 00 70 00 00 00	varSizeValue F.o.l.l. o.w..u. p...
0B 00 4D 81-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 0E 85 00 00	propDef **PidLidAgingDontAgeMe** property (0x850E [PSETID_Common]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
03 00 84 81-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 18 85 00 00	propDef **PidLidTaskMode** property (0x8518 [PSETID_Common]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
0B 00 4B 82-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 06 85 00 00	propDef **PidLidPrivate** property (0x8506 [PSETID_Common]) [Bool]
00 00	fixedSizeValue [Bool] FALSE

Bytes on the wire	Value/description
0B 00 4F 82-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 82 85 00 00	propDef **PidLidUseTnef** property (0x8582 [PSETID_Common]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
40 00 68 82-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 A0 85 00 00	propDef **PidLidToDoOrdinalDate** property ([MS-OXOFLAG] section 2.2.1.13) (0x85A0 [PSETID_Common]) [SysTime]
F0 55 C3 C6-8B 84 C8 01	fixedSizeValue [SysTime] 2008-03-12T21:55:25.0070000
1F 00 69 82-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 A1 85 00 00	propDef **PidLidToDoSubOrdinal** property ([MS-OXOFLAG] section 2.2.1.14) (0x85A1 [PSETID_Common]) [Unicode]
10 00 00 00	length 16 (0x10)
35 00 35 00-35 00 35 00 35 00 35 00-35 00 00 00	varSizeValue 5.5.5.5. 5.5.5...
03 00 A8 83-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 01 85 00 00	propDef **PidLidReminderDelta** property (0x8501 [PSETID_Common]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
40 00 A9 83-03 20 06 00 00 00 00 00-	propDef **PidLidTaskDueDate** property ([MS-OXOTASK] section 2.2.2.2.5) (0x8105 [PSETID_Task]) [SysTime]

Bytes on the wire	Value/description
C0 00 00 00 00 00 00 46-00 05 81 00 00	
00 00 CB 03-D4 83 C8 01	fixedSizeValue [SysTime] 2008-03-12T00:00:00.0000000
40 00 AA 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 04 81 00 00	propDef **PidLidTaskStartDate** property ([MS-OXOTASK] section 2.2.2.2.4) (0x8104 [PSETID_Task]) [SysTime]
00 00 CB 03-D4 83 C8 01	fixedSizeValue [SysTime] 2008-03-12T00:00:00.0000000
40 00 AB 83-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 16 85 00 00	propDef **PidLidCommonStart** property ([MS-OXOTASK] section 2.2.2.1.3) (0x8516 [PSETID_Common]) [SysTime]
00 D8 29 B0-0E 84 C8 01	fixedSizeValue [SysTime] 2008-03-12T07:00:00.0000000
40 00 AC 83-08 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 17 85 00 00	propDef **PidLidCommonEnd** property ([MS-OXOTASK] section 2.2.2.1.4) (0x8517 [PSETID_Common]) [SysTime]
00 D8 29 B0-0E 84 C8 01	fixedSizeValue [SysTime] 2008-03-12T07:00:00.0000000
03 00 AD 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 01 81 00 00	propDef **PidLidTaskStatus** property ([MS-OXOTASK] section 2.2.2.2.2) (0x8101 [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0

Bytes on the wire	Value/description
05 00 AE 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 02 81 00 00	propDef **PidLidPercentComplete** property (0x8102 [PSETID_Task]) [Double]
00 00 00 00-00 00 00 00	fixedSizeValue [Double] 0
0B 00 B0 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 1C 81 00 00	propDef **PidLidTaskComplete** property (0x811C [PSETID_Task]) [Bool]
00 00	fixedSizeValue [Bool] FALSE
03 00 CA 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 13 81 00 00	propDef **PidLidTaskState** property (0x8113 [PSETID_Task]) [Int32]
01 00 00 00	fixedSizeValue [Int32] 1
03 00 CB 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 12 81 00 00	propDef **PidLidTaskVersion** property (0x8112 [PSETID_Task]) [Int32]
01 00 00 00	fixedSizeValue [Int32] 1
03 00 CC 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 23 81 00 00	propDef **PidLidTaskOrdinal** property (0x8123 [PSETID_Task]) [Int32]

Bytes on the wire	Value/description
FF FF FF 7F	fixedSizeValue [Int32] 2147483647
1F 00 CD 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 21 81 00 00	propDef **PidLidTaskAssigner** property (0x8121 [PSETID_Task]) [Unicode]
02 00 00 00	length 2 (0x2)
00 00	varSizeValue ..
03 00 CE 83-03 20 06 00 00 00 00 00-C0 00 00 00 00 00 00 46-00 29 81 00 00	propDef **PidLidTaskOwnership** property (0x8129 [PSETID_Task]) [Int32]
00 00 00 00	fixedSizeValue [Int32] 0
03 00 03 40	**marker** **PidTagStartRecip** marker (section 2.2.4.1.4) (40030003 [Int32])
03 00 00 30	propDef **PidTagRowid** property (30000003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
1F 00 02 30	propDef **PidTagAddressType** property (3002001F [Unicode])
06 00 00 00	length 6 (0x6)
45 00 58 00-00 00	varSizeValue E.X...
1F 00 03 30	propDef **PidTagEmailAddress** property (3003001F [Unicode])
BA 00 00 00	length 186 (0xBA)

Bytes on the wire	Value/description
2F 00 4F 00-3D 00 46 00 49 00 52 00-53 00 54 00 20 00 4F 00-52 00 47 00 41 00 4E 00-49 00 5A 00 41 00 54 00-49 00 4F 00	varSizeValue /.O.=.F. I.R.S.T. .O.R.G. A.N.I.Z. A.T.I.O.
... value truncated...	
1F 00 01 30	propDef **PidTagDisplayName** property (3001001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
02 01 F6 0F	propDef **PidTagInstanceKey** property (0FF60102 [Binary])
04 00 00 00	length 4 (0x4)
00 00 00 00	varSizeValue ....
03 00 15 0C	propDef **PidTagRecipientType** property (0C150003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
02 01 FF 0F	propDef **PidTagEntryId** property (0FFF0102 [Binary])
79 00 00 00	length 121 (0x79)
00 00 00 00-DC A7 40 C8 C0 42 10 1A-B4 B9 08 00 2B 2F E1 82-01 00 00 00 00 00 00 00-2F 4F 3D 46 49 52 53 54-	varSizeValue .....@. .B..... +/..... ..../O=F IRST ORG

Bytes on the wire	Value/description
20 4F 52 47	
... value truncated...	
02 01 0B 30	propDef **PidTagSearchKey** property (300B0102 [Binary])
60 00 00 00	length 96 (0x60)
45 58 3A 2F- 4F 3D 46 49 52 53 54 20- 4F 52 47 41 4E 49 5A 41- 54 49 4F 4E 2F 4F 55 3D- 45 58 43 48 41 4E 47 45- 20 41 44 4D	varSizeValue EX:/O=FI RST ORGA NIZATION /OU=EXCH ANGE ADM
... value truncated...	
1F 00 20 3A	propDef **PidTagTransmittableDisplayName** property (3A20001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00- 00 00	varSizeValue t.1...
0B 00 0F 0E	propDef **PidTagResponsibility** property (0E0F000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
0B 00 40 3A	propDef **PidTagSendRichInfo** property (3A40000B [Bool])
01 00	fixedSizeValue [Bool] TRUE
03 00 FD 5F	propDef **PidTagRecipientFlags** property (5FFD0003 [Int32])
01 00 00 00	fixedSizeValue [Int32] 1
02 01 F7 5F	propDef **PidTagRecipientEntryId** property (5FF70102 [Binary])

Bytes on the wire	Value/description
79 00 00 00	length 121 (0x79)
00 00 00 00- DC A7 40 C8 C0 42 10 1A- B4 B9 08 00 2B 2F E1 82- 01 00 00 00 00 00 00 00- 2F 6F 3D 46 69 72 73 74- 20 4F 72 67	varSizeValue .....@. .B..... +/..... ..../o=F irst Org
... value truncated...	
1F 00 FE 39	propDef **PidTagPrimarySmtpAddress** property (39FE001F [Unicode])
46 00 00 00	length 70 (0x46)
74 00 31 00- 40 00 65 00 75 00 6D 00- 61 00 72 00 75 00 2D 00- 64 00 6F 00 6D 00 2E 00- 65 00 78 00 74 00 65 00- 73 00 74 00	varSizeValue t.1.@.e. u.m.a.r. u.-.d.o. m...e.x. t.e.s.t.
... value truncated...	
03 00 05 39	propDef **PidTagDisplayTypeEx** property (39050003 [Int32])
00 00 00 40	fixedSizeValue [Int32] 1073741824
03 00 00 39	propDef **PidTagDisplayType** property (39000003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 FE 0F	propDef **PidTagObjectType** property (0FFE0003 [Int32])
06 00 00 00	fixedSizeValue [Int32] 6

Bytes on the wire	Value/description
1F 00 FF 39	propDef **PidTag7BitDisplayName** property (39FF001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
1F 00 00 3A	propDef **PidTagAccount** property (3A00001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
03 00 DE 5F	propDef **PidTagRecipientResourceState** property (5FDE0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 DF 5F	propDef **PidTagRecipientOrder** property (5FDF0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
1F 00 F6 5F	propDef **PidTagRecipientDisplayName** property (5FF6001F [Unicode])
06 00 00 00	length 6 (0x6)
74 00 31 00-00 00	varSizeValue t.1...
03 00 FF 5F	propDef **PidTagRecipientTrackStatus** property (5FFF0003 [Int32])
00 00 00 00	fixedSizeValue [Int32] 0
03 00 04 40	**marker** **PidTagEndToRecip** marker (section 2.2.4.1.4) (40040003 [Int32])
03 00 02 40	**marker** **PidTagEndEmbed** marker (section 2.2.4.1.4) (40020003 [Int32])

Bytes on the wire	Value/description
03 00 0E 40	**marker** **PidTagEndAttach** marker (section 2.2.4.1.4) (400E0003 [Int32])
03 00 13 40	**marker** **PidTagIncrSyncDel** marker (section 2.2.4.1.4) (40130003 [Int32])
02 01 E5 67	propDef **PidTagIdsetDeleted** property (section 2.2.1.3.1) (67E50102 [Binary])
0D 00 00 00	length 13 (0xD)
01 00 06 00-00 00 78 2E 23 00 04 00-00	varSizeValue .....x. #....
03 00 2F 40	**marker** **PidTagIncrSyncRead** marker (section 2.2.4.1.4) (402F0003 [Int32])
02 01 2D 40	propDef **PidTagIdsetRead** property (section 2.2.1.3.4) (402D0102 [Binary])
0A 00 00 00	length 10 (0xA)
01 00 06 00-00 00 78 2E 1F 00	varSizeValue .....x. ..
02 01 2E 40	propDef **PidTagIdsetUnread** property (section 2.2.1.3.5) (402E0102 [Binary])
0A 00 00 00	length 10 (0xA)
01 00 06 00-00 00 78 2E 20 00	varSizeValue .....x. .
03 00 3A 40	**marker** **PidTagIncrSyncStateBegin** marker (section 2.2.4.1.4) (403A0003 [Int32])
02 01 96 67	propDef **PidTagCnsetSeen** property (section 2.2.1.1.2) (67960102 [Binary])
1D 00 00 00	length 29 (0x1D)
19 D7 FB 0F-06 16 A1 41 BF F6 91 C7-	IDSET printout: {0ffbd719-1606-41a1-bff6-91c763daa866:{[0x1, 0x784D1D]}}

Bytes on the wire	Value/description
63 DA A8 66 03 00 00 00-52 00 00 01 78 4D 1D 50-00	
02 01 DA 67	propDef **PidTagCnsetSeenFAI** (section 2.2.1.1.3) (67DA0102 [Binary])
1D 00 00 00	length 29 (0x1D)
19 D7 FB 0F-06 16 A1 41 BF F6 91 C7-63 DA A8 66 03 00 00 00-52 00 00 01 78 4D 1D 50-00	IDSET printout: {0ffbd719-1606-41a1-bff6-91c763daa866:{[0x1, 0x784D1D]}}
03 00 17 40	propDef **PidTagIdsetGiven** property (section 2.2.1.1.1) (40170003 [Int32])
38 00 00 00	length 56 (0x38)
19 D7 FB 0F-06 16 A1 41 BF F6 91 C7-63 DA A8 66 05 00 00 00-78 2E 52 1D 22 50 00 D2-0C 67 79 AC 4C 50 42 89-2C 24 5D 2D 1A E3 A4 05-00 00 00 78 06 42 01 80-01 0C 50 00	IDSET printout: {0ffbd719-1606-41a1-bff6-91c763daa866:{[0x782E1D, 0x782E22]},79670cd2-4cac-4250-892c-245d2d1ae3a4:{[0x780601, 0x780602], [0x78060C, 0x78060C]}}
02 01 D2 67	propDef **PidTagCnsetRead** property (section 2.2.1.1.4) (67D20102 [Binary])
1D 00 00 00	length 29 (0x1D)
19 D7 FB 0F-06 16 A1 41 BF F6 91 C7-63 DA A8 66	IDSET printout: {0ffbd719-1606-41a1-bff6-91c763daa866:{[0x1, 0x784D1D]}}

Bytes on the wire	Value/description
03 00 00 00- 52 00 00 01 78 4D 1D 50-00	
03 00 3B 40	**marker** **PidTagIncrSyncStateEnd** marker (section 2.2.4.1.4) (403B0003 [Int32])
03 00 14 40	**marker** **PidTagIncrSyncEnd** marker (section 2.2.4.1.4) (40140003 [Int32])
	EOS

5 Security

5.1 Security Considerations for Implementers

Individual security considerations are specified in section 3.2.5.5.4.7 and section 3.2.5.5.4.8.

There are no additional security considerations specific to the Bulk Data Transfer Protocol. Security considerations pertaining to the underlying Wire Format Protocol, as described in [MS-OXCRPC] section 5.1, do apply to this specification.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® Exchange Server 2003
- Microsoft® Exchange Server 2007
- Microsoft® Exchange Server 2010
- Microsoft® Exchange Server 2010 Service Pack 1 (SP1)
- Microsoft® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010

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

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

<1> Section 2.2.1.7: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not use the **PidTagContainerHierarchy** property ([MS-OXPROPS] section 2.715) as a **PidTagFXDelProp** meta-property (section 2.2.4.1.5.1).

<2> Section 2.2.1.7: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not use the **PidTagAttachDataObject** property ([MS-OXCMSG] section 2.2.2.8) as a **PidTagFXDelProp** meta-property.

<3> Section 2.2.3: In Exchange 2007, the **RopSynchronizationGetTransferState** ([MS-OXCROPS] section 2.2.13.8) and the **RopFastTransferSourceGetBuffer** ([MS-OXCROPS] section 2.2.12.3) ROPs are used to checkpoint ICS download operations. In Exchange 2010, trying to retrieve a checkpoint ICS state during the download will return the initial state, and not a state with differences applied.

<4> Section 2.2.3.1.1.1.1: Exchange 2010 does not support the **Move** flag for the **RopFastTransferSourceCopyTo** ROP (section 2.2.3.1.1.1). The server sends a **ReturnValue** of **InvalidParameter** (0x80070057) if it receives this flag.

<5> Section 2.2.3.1.1.1.1: Exchange 2003 and Office Outlook 2003 do not support partial item downloads. Exchange 2003 does not recognize the **SendOptions** flag **PartialItem** and Office Outlook 2003 does not pass it.

<6> Section 2.2.3.1.1.4.1: In Exchange 2003 and Exchange 2007, the **Move** flag is not ignored by the server.

<7> Section 2.2.3.1.1.5.2: Exchange 2003 and Exchange 2007 have an additional *TransferStatus* value, **NoRoom** ("0x0002"). The **NoRoom** value indicates that the FastTransfer stream was split,

more data is available, and **TransferBuffer** contains incomplete data. Note that **NoRoom** is not returned by Exchange 2010.

<8> Section 2.2.3.1.1.5.2: Exchange 2010 reports inaccurate information in this output parameter when client connection services are deployed on an Microsoft Exchange that does not also have a mailbox store installed.

<9> Section 2.2.3.2.1.1.1: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not implement the **NoDeletions** flag.

<10> Section 2.2.3.2.1.1.1: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not implement the **IgnoreNoLongerInScope** flag.

<11> Section 2.2.3.2.1.1.1: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not implement the **Reserved** flag.

<12> Section 2.2.3.2.3.1: In Exchange 2007, the RopSynchronizationGetTransferState ROP (section 2.2.3.2.3.1) returns a checkpoint ICS state that is reflective of the current snapshot.

<13> Section 2.2.3.2.4.2.1: The FailOnConflict flag is not supported by Exchange 2003, Exchange 2007, or the initial release version of Exchange 2010.

<14> Section 2.2.3.2.4.5.1: The HardDelete flag is not supported by Exchange 2003 or Exchange 2007.

<15> Section 2.2.4.1.3: Exchange 2003 and Exchange 2007 fail to add a null-terminator when string values are larger than 32K.

<16> Section 2.2.4.3.15: Exchange 2003 and Office Outlook 2003 do not support partial item downloads. Exchange 2003 does not recognize the **SendOptions** flag **PartialItem** and Office Outlook 2003 does not pass it.

<17> Section 3.1.5.4: Exchange 2003 and Office Outlook 2003 do not support partial item downloads. Exchange 2003 does not recognize the SendOptions flag PartialItem and Office Outlook 2003 does not pass it.

<18> Section 3.1.5.5.3.2: Exchange 2010 sends an **RpcFormat** error (0x000004B6), and Exchange 2003 and Exchange 2007 send a **FormatError** error (0x000004ED), as specified in [MS-OXCDATA] section 2.4.1, if the server encounters an unsupported command or any other decoding failures.

<19> Section 3.1.5.5.3.2.3: Exchange 2010 sends an **RpcFormat** error (0x000004B6), and Exchange 2003 and Exchange 2007 send a **FormatError** error (0x000004ED), as specified in [MS-OXCDATA] section 2.4.1, if the server encounters the **Bitmask** command when there are more or fewer than five bytes in the common byte stack.

<20> Section 3.1.5.5.3.2.4: Exchange 2010 sends an RpcFormat error (0x000004B6) and Exchange 2003 and Exchange 2007 send a FormatError error (0x000004ED), as specified in [MS-OXCDATA] section 2.4.1, if the if the high value of the range is larger than the low value of the range.

<21> Section 3.1.5.8: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not use folderContent as a root element in a fast transfer stream.

<22> Section 3.1.5.8: The **RopFastTransferSourceCopyProperties** ROP (section 2.2.3.1.1.2) does not use attachmentContent as a root element in a FastTransfer stream in Office Outlook 2003, Office Outlook 2007, or Outlook 2010.

<23> Section 3.2.5.1: In Exchange 2007, the server confirms that the FastTransfer context that is returned by the **RopSynchronizationGetTransferState** ROP (section 2.2.3.2.3.1), which is sent before the subsequent **RopFastTransferSourceGetBuffer** ROP (section 2.2.3.1.1.5), contains only the differences that have been downloaded to the client in the current synchronization download operation, in addition to what was reflected in the initial ICS state. Note that the final ICS state that has to be downloaded in the FastTransfer stream as the last portion of the payload is exactly the same as the checkpoint ICS state that corresponds to the end of the operation.

<24> Section 3.2.5.5.1.1: Exchange 2003, Exchange 2007, and Exchange 2010 do not honor this flag for embedded messages included in the synchronization scope of a **RopSynchronizationConfigure** ROP request.

<25> Section 3.2.5.5.1.1: Exchange 2003, Exchange 2007, and Exchange 2010 define additional flags for this enumeration, which are only used in server-to-server communications. For that reason, the ROP will not fail if those flags are passed from clients.

<26> Section 3.3.5.3: In Exchange 2003 and Exchange 2007, synchronization download operations function the same as synchronization upload operations. In Exchange 2003 and Exchange 2007, the server returns checkpoint ICS states that are accurate to the time at which the checkpoint was requested in both synchronization download operations and synchronization upload operations.

<27> Section 3.3.5.4.1.5: In Exchange 2003 and Exchange 2007, clients are not required to pass a **BufferSize** value of a certain size.

<28> Section 3.3.5.4.1.5: Office Outlook 2003 does not recognize the **ServerBusy** error code.

<29> Section 3.3.5.5.3.1: In Exchange 2003 and Exchange 2007, the checkpoint ICS state that is returned by the **RopSynchronizationGetTransferState** ROP is accurate to the time at which checkpoint was requested.

<30> Section 4.5: Exchange 2003 and Office Outlook 2003 do not support partial item downloads. Exchange 2003 does not recognize the **SendOptions** flag **PartialItem** and Office Outlook 2003 does not pass it.

7 Change Tracking

This section identifies changes that were made to the [MS-OXCFXICS] protocol document between the November 2010 and March 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

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

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

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

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1 Introduction	Added information about which sections of the specification are normative and can contain RFC 2119 language.	Y	New content added for template compliance.
1.1 Glossary	Added and defined the terms "Camel-cased" and "Pascal-cased".	N	Content updated.
2.2 Message Syntax	Added a description for "No restrictions" in the Restriction column.	N	Content updated.
2.2.1.1.1 PidTagIdsetGiven	Updated the description to state that the property uses a REPLGUID-based IDSET.	N	Content updated.
2.2.1.1.2 PidTagCnsetSeen	Updated the description to state that the property uses a REPLGUID-based IDSET.	N	Content updated.
2.2.1.1.3 PidTagCnsetSeenFAI	Updated the description to state that the property uses a REPLGUID-based IDSET.	N	Content updated.
2.2.1.1.4 PidTagCnsetRead	Updated the description to state that the property uses a REPLGUID-based IDSET.	N	Content updated.
2.2.2.1 CN	Added a section that defines the CN structure.	N	Content updated.
2.2.2.4	Changed the name of the section to	N	Content

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
IDSET and CNSET	"IDSET and CNSET" and added more details about CNSETs.		updated.
2.2.2.4.1 Serialized IDSET with REPLID	Removed glossary link from "formatted IDSET".	N	Content updated.
2.2.2.5 GLOBSET	Removed the word "typically" from the GLOBSET definition.	N	Content updated.
2.2.3.1.1.1 RopFastTransferSourceCopyTo ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.1.1.1 RopFastTransferSourceCopyTo ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.1.1.1 Request Buffer	Added a section that defines the RopFastTransferSourceCopyTo request buffer.	N	Content updated.
2.2.3.1.1.1.1 Request Buffer	Moved request buffer details from the RopFastTransferSourceCopyTo section.	N	Content updated.
2.2.3.1.1.1.2 Response Buffer	Added a section that defines the RopFastTransferSourceCopyTo response buffer.	N	Content updated.
2.2.3.1.1.1.2 Response Buffer	Moved response buffer details from the RopFastTransferSourceCopyTo section.	N	Content updated.
2.2.3.1.1.2 RopFastTransferSourceCopyProperties ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.1.1.2 RopFastTransferSourceCopyProperties ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.1.2.1 Request Buffer	Added a section that defines the RopFastTransferSourceCopyProperties request buffer.	N	Content updated.
2.2.3.1.1.2.1 Request Buffer	Moved request buffer details from the RopFastTransferSourceCopyProperties section.	N	Content updated.
2.2.3.1.1.2.2 Response Buffer	Added a section that defines the RopFastTransferSourceCopyProperties response buffer.	N	Content updated.
2.2.3.1.1.2.2 Response Buffer	Moved response buffer details from the RopFastTransferSourceCopyProperties section.	N	Content updated.
2.2.3.1.1.3 RopFastTransferSourceCopyMessages	Moved request buffer details to the Request Buffer section.	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
ROP			
2.2.3.1.1.3 RopFastTransferSourceCopyMessages ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.1.3.1 Request Buffer	Added a section that defines the RopFastTransferSourceCopyMessages request buffer.	N	Content updated.
2.2.3.1.1.3.1 Request Buffer	Moved request buffer details from the RopFastTransferSourceCopyMessages section.	N	Content updated.
2.2.3.1.1.3.2 Response Buffer	Added a section that defines the RopFastTransferSourceCopyMessages response buffer.	N	Content updated.
2.2.3.1.1.3.2 Response Buffer	Moved response buffer details from the RopFastTransferSourceCopyMessages section.	N	Content updated.
2.2.3.1.1.4 RopFastTransferSourceCopyFolder ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.1.1.4 RopFastTransferSourceCopyFolder ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.1.4.1 Request Buffer	Added a section that defines the RopFastTransferSourceCopyFolder request buffer.	N	Content updated.
2.2.3.1.1.4.1 Request Buffer	Moved request buffer details from the RopFastTransferSourceCopyFolder section.	N	Content updated.
2.2.3.1.1.4.2 Response Buffer	Added a section that defines the RopFastTransferSourceCopyFolder response buffer.	N	Content updated.
2.2.3.1.1.4.2 Response Buffer	Moved response buffer details from the RopFastTransferSourceCopyFolder section.	N	Content updated.
2.2.3.1.1.5 RopFastTransferSourceGetBuffer ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.1.1.5 RopFastTransferSourceGetBuffer ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.1.5.1 Request Buffer	Added a section that defines the RopFastTransferSourceGetBuffer request buffer.	N	Content updated.
2.2.3.1.1.5.1	Moved request buffer details from the	N	Content

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
Request Buffer	RopFastTransferSourceGetBuffer section.		updated.
2.2.3.1.1.5.2 Response Buffer	Added a section that defines the RopFastTransferSourceGetBuffer response buffer.	N	Content updated.
2.2.3.1.1.5.2 Response Buffer	Moved response buffer details from the RopFastTransferSourceGetBuffer section.	N	Content updated.
2.2.3.1.1.6 RopTellVersion ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.1.1.6 RopTellVersion ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.1.6.1 Request Buffer	Added a section that defines the RopTellVersion request buffer.	N	Content updated.
2.2.3.1.1.6.1 Request Buffer	Moved request buffer details from the RopTellVersion section.	N	Content updated.
2.2.3.1.1.6.2 Response Buffer	Added a section that defines the RopTellVersion response buffer.	N	Content updated.
2.2.3.1.1.6.2 Response Buffer	Moved response buffer details from the RopTellVersion section.	N	Content updated.
2.2.3.1.2.1 RopFastTransferDestinationConfigure ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.1.2.1 RopFastTransferDestinationConfigure ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.2.1.1 Request Buffer	Added a section that defines the RopFastTransferDestinationConfigure request buffer.	N	Content updated.
2.2.3.1.2.1.1 Request Buffer	Moved request buffer details from the RopFastTransferDestinationConfigure section.	N	Content updated.
2.2.3.1.2.1.2 Response Buffer	Added a section that defines the RopFastTransferDestinationConfigure response buffer.	N	Content updated.
2.2.3.1.2.1.2 Response Buffer	Moved response buffer details from the RopFastTransferDestinationConfigure section.	N	Content updated.
2.2.3.1.2.2 RopFastTransferDestinationPutBuffer ROP	Added details about the maximum size of the TransferData parameter.	Y	Content updated.
2.2.3.1.2.2	Moved request buffer details to the	N	Content

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
RopFastTransferDestinationPutBuffer ROP	Request Buffer section.		updated.
2.2.3.1.2.2 RopFastTransferDestinationPutBuffer ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.1.2.2.1 Request Buffer	Added a section that defines the RopFastTransferDestinationPutBuffer request buffer.	N	Content updated.
2.2.3.1.2.2.1 Request Buffer	Moved request buffer details from the RopFastTransferDestinationPutBuffer section.	N	Content updated.
2.2.3.1.2.2.2 Response Buffer	Added a section that defines the RopFastTransferDestinationPutBuffer response buffer.	N	Content updated.
2.2.3.1.2.2.2 Response Buffer	Moved response buffer details from the RopFastTransferDestinationPutBuffer section.	N	Content updated.
2.2.3.2.1.1 RopSynchronizationConfigure ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.1.1 RopSynchronizationConfigure ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.1.1.1 Request Buffer	Added a section that defines the RopSynchronizationConfigure request buffer.	N	Content updated.
2.2.3.2.1.1.1 Request Buffer	Moved request buffer details from the RopSynchronizationConfigure section.	N	Content updated.
2.2.3.2.1.1.2 Response Buffer	Added a section that defines the RopSynchronizationConfigure response buffer.	N	Content updated.
2.2.3.2.1.1.2 Response Buffer	Moved response buffer details from the RopSynchronizationConfigure section.	N	Content updated.
2.2.3.2.2.1 RopSynchronizationUploadStateStreamBegin ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.2.1 RopSynchronizationUploadStateStreamBegin ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.2.1 RopSynchronizationUploadStateStreamBegin ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.2.1.1 Request Buffer	Added a section that defines the RopSynchronizationUploadStateStreamBe	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
	gin request buffer.		
2.2.3.2.2.1.1 Request Buffer	Moved request buffer details from the RopSynchronizationUploadStateStreamBegin section.	N	Content updated.
2.2.3.2.2.1.2 Response Buffer	Added a section that defines the RopSynchronizationUploadStateStreamBegin response buffer.	N	Content updated.
2.2.3.2.2.1.2 Response Buffer	Moved response buffer details from the RopSynchronizationUploadStateStreamBegin section.	N	Content updated.
2.2.3.2.2.2 RopSynchronizationUploadStateStreamContinue ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.2.2 RopSynchronizationUploadStateStreamContinue ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.2.2.1 Request Buffer	Added a section that defines the RopSynchronizationUploadStateStreamContinue request buffer.	N	Content updated.
2.2.3.2.2.2.1 Request Buffer	Moved request buffer details from the RopSynchronizationUploadStateStreamContinue section.	N	Content updated.
2.2.3.2.2.2.2 Response Buffer	Added a section that defines the RopSynchronizationUploadStateStreamContinue response buffer.	N	Content updated.
2.2.3.2.2.2.2 Response Buffer	Moved response buffer details from the RopSynchronizationUploadStateStreamContinue section.	N	Content updated.
2.2.3.2.2.3 RopSynchronizationUploadStateStreamEnd ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.2.3 RopSynchronizationUploadStateStreamEnd ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.2.3.1 Request Buffer	Added a section that defines the RopSynchronizationUploadStateStreamEnd request buffer.	N	Content updated.
2.2.3.2.2.3.1 Request Buffer	Moved request buffer details from the RopSynchronizationUploadStateStreamEnd section.	N	Content updated.
2.2.3.2.2.3.2	Added a section that defines the	N	Content

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
Response Buffer	RopSynchronizationUploadStateStreamEnd response buffer.		updated.
2.2.3.2.2.3.2 Response Buffer	Moved response buffer details from the RopSynchronizationUploadStateStreamEnd section.	N	Content updated.
2.2.3.2.3.1 RopSynchronizationGetTransferState ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.3.1 RopSynchronizationGetTransferState ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.3.1.1 Request Buffer	Added a section that defines the RopSynchronizationGetTransferState request buffer.	N	Content updated.
2.2.3.2.3.1.1 Request Buffer	Moved request buffer details from the RopSynchronizationGetTransferState section.	N	Content updated.
2.2.3.2.3.1.2 Response Buffer	Added a section that defines the RopSynchronizationGetTransferState response buffer.	N	Content updated.
2.2.3.2.3.1.2 Response Buffer	Moved response buffer details from the RopSynchronizationGetTransferState section.	N	Content updated.
2.2.3.2.4.1 RopSynchronizationOpenCollector ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.4.1 RopSynchronizationOpenCollector ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.1.1 Request Buffer	Added a section that defines the RopSynchronizationOpenCollector request buffer.	N	Content updated.
2.2.3.2.4.1.1 Request Buffer	Moved request buffer details from the RopSynchronizationOpenCollector section.	N	Content updated.
2.2.3.2.4.1.2 Response Buffer	Added a section that defines the RopSynchronizationOpenCollector response buffer.	N	Content updated.
2.2.3.2.4.1.2 Response Buffer	Moved response buffer details from the RopSynchronizationOpenCollector section.	N	Content updated.
2.2.3.2.4.2 RopSynchronizationImportMessageChange	Moved request buffer details to the Request Buffer section.	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.3.2.4.2 RopSynchronizationImportMessageChange	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.2.1 Request Buffer	Added a section that defines the RopSynchronizationImportMessageChange request buffer.	N	Content updated.
2.2.3.2.4.2.1 Request Buffer	Moved request buffer details from the RopSynchronizationImportMessageChange section.	N	Content updated.
2.2.3.2.4.2.2 Response Buffer	Added a section that defines the RopSynchronizationImportMessageChange response buffer.	N	Content updated.
2.2.3.2.4.2.2 Response Buffer	Moved response buffer details from the RopSynchronizationImportMessageChange section.	N	Content updated.
2.2.3.2.4.3 RopSynchronizationImportHierarchyChange ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.4.3 RopSynchronizationImportHierarchyChange ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.3.1 Request Buffer	Added a section that defines the RopSynchronizationImportHierarchyChange request buffer.	N	Content updated.
2.2.3.2.4.3.1 Request Buffer	Moved request buffer details from the RopSynchronizationImportHierarchyChange section.	N	Content updated.
2.2.3.2.4.3.2 Response Buffer	Added a section that defines the RopSynchronizationImportHierarchyChange response buffer.	N	Content updated.
2.2.3.2.4.3.2 Response Buffer	Moved response buffer details from the RopSynchronizationImportHierarchyChange section.	N	Content updated.
2.2.3.2.4.4 RopSynchronizationImportMessageMove ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.4.4 RopSynchronizationImportMessageMove ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.4.1 Request Buffer	Added a section that defines the RopSynchronizationImportMessageMove request buffer.	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.3.2.4.4.1 Request Buffer	Moved request buffer details from the RopSynchronizationImportMessageMove section.	N	Content updated.
2.2.3.2.4.4.2 Response Buffer	Added a section that defines the RopSynchronizationImportMessageMove response buffer.	N	Content updated.
2.2.3.2.4.4.2 Response Buffer	Moved response buffer details from the RopSynchronizationImportMessageMove section.	N	Content updated.
2.2.3.2.4.5 RopSynchronizationImportDeletes ROP	Added the PropertyValueCount field to the request buffer.	Y	Content updated.
2.2.3.2.4.5 RopSynchronizationImportDeletes ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.4.5 RopSynchronizationImportDeletes ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.5.1 Request Buffer	Added a section that defines the RopSynchronizationImportDeletes request buffer.	N	Content updated.
2.2.3.2.4.5.1 Request Buffer	Moved request buffer details from the RopSynchronizationImportDeletes section.	N	Content updated.
2.2.3.2.4.5.2 Response Buffer	Added a section that defines the RopSynchronizationImportDeletes response buffer.	N	Content updated.
2.2.3.2.4.5.2 Response Buffer	Moved response buffer details from the RopSynchronizationImportDeletes section.	N	Content updated.
2.2.3.2.4.6 RopSynchronizationImportReadStateChanges ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.4.6 RopSynchronizationImportReadStateChanges ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.6.1 Request Buffer	Moved request buffer details from the RopSynchronizationImportReadStateChanges section.	N	Content updated.
2.2.3.2.4.6.1 Request Buffer	Added a section that defines the RopSynchronizationImportReadStateChanges request buffer.	N	Content updated.
2.2.3.2.4.6.2 Response Buffer	Added a section that defines the RopSynchronizationImportReadStateCha	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
	nges response buffer.		
2.2.3.2.4.7 RopGetLocalReplicaIds ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.4.7 RopGetLocalReplicaIds ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.7.1 Request Buffer	Moved request buffer details from the RopGetLocalReplicaIds section.	N	Content updated.
2.2.3.2.4.7.1 Request Buffer	Added a section that defines the RopGetLocalReplicaIds request buffer.	N	Content updated.
2.2.3.2.4.7.2 Response Buffer	Moved response buffer details from the RopGetLocalReplicaIds section.	N	Content updated.
2.2.3.2.4.7.2 Response Buffer	Added a section that defines the RopGetLocalReplicaIds response buffer.	N	Content updated.
2.2.3.2.4.8 RopSetLocalReplicaMidsetDeleted ROP	Moved request buffer details to the Request Buffer section.	N	Content updated.
2.2.3.2.4.8 RopSetLocalReplicaMidsetDeleted ROP	Moved response buffer details to the Response Buffer section.	N	Content updated.
2.2.3.2.4.8.1 Request Buffer	Moved request buffer details from the RopSetLocalReplicaMidsetDeleted section.	N	Content updated.
2.2.3.2.4.8.1 Request Buffer	Added a section that defines the RopSetLocalReplicaMidsetDeleted request buffer.	N	Content updated.
2.2.3.2.4.8.2 Response Buffer	Moved response buffer details from the RopSetLocalReplicaMidsetDeleted section.	N	Content updated.
2.2.3.2.4.8.2 Response Buffer	Added a section that defines the RopSetLocalReplicaMidsetDeleted response buffer.	N	Content updated.
2.2.4.1.4 Markers	Added descriptions of all the markers.	N	Content updated.
2.2.4.1.5.2 PidTagEcWarning	Changed "client SHOULD NOT assume" to "client SHOULD verify".	N	Content updated.
3.1.5.1 Isolating Download and Upload Operations	Reworded first sentence to remove "MUST NOT assume".	N	Content updated.
3.1.5.3 Identifying Objects and Maintaining Change Numbers	Changed SourceKey to PidTagSourceKey.	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.1.5.3 Identifying Objects and Maintaining Change Numbers	Removed ambiguous language that described altering a change number without flagging it for synchronization.	N	Content updated.
3.1.5.3 Identifying Objects and Maintaining Change Numbers	Added a description for the symbols and notations used in the table. Changed the names of the properties in the table to their PidTag names, such as PidTagChangeNumber, and PidTagSourceKey.	N	Content updated.
3.1.5.5.3 GLOBSET Serialization	Added information about when the common byte stack is constructed.	N	Content updated.
3.1.5.5.3.1.3 Bitmask Command (0x42)	Clarified the bit being described.	N	Content updated.
3.1.5.5.3.1.3 Bitmask Command (0x42)	Changed the description of how the bit being set is determined by the GLOBCNT and the StartingValue.	N	Content updated.
3.1.5.5.3.2 Decoding	Added information that the server sends an error if it encounters an unsupported command.	Y	Content updated.
3.1.5.5.3.2 Decoding	Added information that the server sends an error if it encounters an unsupported command.	Y	Content updated.
3.1.5.5.3.2.1 Push Command (0x01 – 0x06)	Added details about the role of the decoder when it encounters the Push command.	N	Content updated.
3.1.5.5.3.2.3 Bitmask Command (0x42)	Added information that the server sends an error if the decoder encounters the Bitmask command when there are more or less than five bytes in the common byte stack.	Y	Content updated.
3.1.5.5.3.2.4 Range Command (0x52)	Added information that the server sends an error if the high value of the range is larger than the low value of the range.	Y	Content updated.
3.2.5.1 Determining What Differences Need to be Downloaded	Clarified the explanation for when both the SynchronizationFlag NoDeletion and IgnoreNoLongerInScope flags are not set.	N	Content updated.
3.2.5.1 Determining What Differences Need to be Downloaded	Removed reference to invariatns for CnsetSeen, CnsetSeenFAI, and CnsetRead.	N	Content updated.
3.2.5.3 Tracking Read State Changes	Clarified that both the server assigns a new read state change number.	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.2.5.4.1.5 Receiving a RopFastTransferSourceGetBuffer Request	Added details about the maximum size of the output RPC buffer.	Y	Content updated.
3.2.5.4.1.5 Receiving a RopFastTransferSourceGetBuffer Request	Added details about the maximum size of the output RPC buffer.	Y	Content updated.
3.2.5.5.1.1 Receiving a RopSynchronizationConfigure Request	Clarified that the progress information injected into the FastTransfer stream is the progressTotal element.	N	Content updated.
3.2.5.5.4.8 Receiving a RopSetLocalReplicaMidsetDeleted Request	Updated the server requirement regarding adding rages of IDs to the deleted items list.	N	Content updated.
3.3.5.1.1 Client-Assigned Internal Identifiers	Added details about why IDs MUST NOT be used for change numbers.	N	Content updated.
3.3.5.1.2 Use Online Mode ROPs	Removed "MUST" from the definition of the online mode ROP approach.	N	Content updated.
3.3.5.2 Determining the Synchronization Scope	Added the resulting behavior if the client does not subdivide work.	N	Content updated.
3.3.5.4.1.6 Sending a RopTellVersion Request	Added requirement that the RopTellVersion ROP MUST be sent before the first RopFastTransferSourceGetBuffer or RopFastTransferDestinationPutBuffer ROP.	N	Content updated.
3.3.5.5.4.7 Sending a RopSynchronizationImportDeletes Request	Clarified the action taken if the client updates the ICS state of the chosen synchronization scope by removing internal identifiers of deleted objects from its PidTagIdsetGiven property.	N	Content updated.
	Removed the section CopyFlags from the section RopFastTransferSourceCopyTo.	N	Content updated.
	Removed the section SendOptions from the section RopFastTransferSourceCopyTo.	N	Content updated.
	Removed the section CopyFlags from the section RopFastTransferSourceCopyProperties.	N	Content updated.
	Removed the section CopyFlags from the section RopFastTransferSourceCopyMessages.	N	Content updated.
	Removed the section CopyFlags from the section	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
	RopFastTransferSourceCopyFolder.		
	Removed the section TransferStatus from the section RopFastTransferSourceGetBuffer.	N	Content updated.
	Removed the section SourceOperation from the section RopFastTransferDestinationConfigure.	N	Content updated.
	Removed the section CopyFlags from the section RopFastTransferDestinationConfigure.	N	Content updated.
	Removed the section SynchronizationType from the section RopSynchronizationConfigure.	N	Content updated.
	Removed the section SynchronizationFlag from the section RopSynchronizationConfigure.	N	Content updated.
	Removed the section SynchronizationType from the section RopSynchronizationConfigure.	N	Content updated.
	Removed the section SynchronizationExtraFlag from the section RopSynchronizationConfigure.	N	Content updated.
	Removed the section PropertyTags from the section RopSynchronizationConfigure.	N	Content updated.
	Removed the section ImportFlag from the section RopSynchronizationImportMessageChange.	N	Content updated.
	Removed the section ImportDeleteFlags from the section Removed the section ImportFlag from the section RopSynchronizationImportMessageChange.	N	Content updated.

8 Index

V