[MS-OXCTABL]:

Table Object Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial Availability.
4/25/2008	0.2	Minor	Revised and updated property names and other technical content.
6/27/2008	1.0	Major	Initial Release.
8/6/2008	1.01	Minor	Revised and edited technical content.
9/3/2008	1.02	Minor	Revised and edited technical content.
12/3/2008	1.03	Minor	Revised and edited technical content.
4/10/2009	2.0	Major	Updated technical content and applicable product releases.
7/15/2009	3.0	Major	Revised and edited for technical content.
11/4/2009	3.1.0	Minor	Updated the technical content.
2/10/2010	3.2.0	Minor	Updated the technical content.
5/5/2010	4.0.0	Major	Updated and revised the technical content.
8/4/2010	5.0	Major	Significantly changed the technical content.
11/3/2010	6.0	Major	Significantly changed the technical content.
3/18/2011	7.0	Major	Significantly changed the technical content.
8/5/2011	7.1	Minor	Clarified the meaning of the technical content.
10/7/2011	7.1	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	8.0	Major	Significantly changed the technical content.
4/27/2012	9.0	Major	Significantly changed the technical content.
7/16/2012	9.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	10.0	Major	Significantly changed the technical content.
2/11/2013	11.0	Major	Significantly changed the technical content.
7/26/2013	11.0	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	12.0	Major	Significantly changed the technical content.
2/10/2014	12.0	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	12.1	Minor	Clarified the meaning of the technical content.
7/31/2014	12.1	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	13.0	Major	Significantly changed the technical content.

Date	Revision History	Revision Class	Comments
3/16/2015	14.0	Major	Significantly changed the technical content.
5/26/2015	15.0	Major	Significantly changed the technical content.
9/14/2015	16.0	Major	Significantly changed the technical content.
6/13/2016	17.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Table Object Protocol provides properties and operations that allow a client to read and navigate through data that is retrieved in tabular format from a server. In addition to retrieving filtered, sorted rows of tabular data, this protocol also allows a client to collapse a grouping of rows and to navigate through those rows.

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

1.1 Glossary

This document uses the following terms:

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

binary large object (BLOB): A discrete packet of data that is stored in a database and is treated as a sequence of uninterpreted bytes.

bookmark: A data structure that the server uses to point to a position in the **Table object**. There are three pre-defined bookmarks (beginning, end, and current). A custom bookmark is a server-specific data structure that can be stored by the client for easily navigating a **Table object**.

category: A grouping of rows in a **Table object** that all have the same value for a specified property.

column set: A set of properties that are requested by a client for each row of data.

contents table: A **Table object** whose rows represent the Message objects that are contained in a **Folder object**.

flags: A set of values used to configure or report options or settings.

Folder object: A messaging construct that is typically used to organize data into a hierarchy of objects containing Message objects and folder associated information (FAI) Message objects.

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

handle array: An array of object handles that are sent to and received from a server as part of a remote procedure call (RPC) accompanying **ROP request buffers** and **ROP response buffers**, respectively. Also referred to as a Server object handle table or an HSOT table.

header row: A row at the beginning of a **category** that does not represent data in the **Table object**, but provides information about a grouping.

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

instance: A unique publication of data for a **category**. It enables a publisher to publish data for the same category multiple times. An example is a publisher who uses two different endpoints (5) to publish data. These endpoints can publish the same category. However, each endpoint requires a different instance number to be considered a distinct publication by the server (2). An instance number is provided by the publishing client.

leaf row: A row that is in a **category**.

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

multivalue instance: A row that is in a table and corresponds to a single value in a multivalue property. There are multiple rows for each Message object in a table and each row corresponds to one value of the multivalue property. Each row has a single value for the property and the properties for the other columns are repeated.

multivalue property: A property that can contain multiple values of the same type.

permission: A rule that is associated with an object and that regulates which users can gain access to the object and in what manner. See also rights.

permissions table: A **Table object** whose rows represent entries in a permissions list for a **Folder object**.

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

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

property type: A 16-bit quantity that specifies the data type of a property value.

remote operation (ROP): An operation that is invoked against a server. Each ROP represents an action, such as delete, send, or query. A ROP is contained in a ROP buffer for transmission over the wire.

restriction: A filter used to map some domain into a subset of itself, by passing only those items from the domain that match the filter. Restrictions can be used to filter existing **Table objects** or to define new ones, such as search folder (2) or rule criteria.

ROP request: See **ROP request buffer**.

ROP request buffer: A ROP buffer that a client sends to a server to be processed.

ROP response: See **ROP response buffer**.

ROP response buffer: A ROP buffer that a server sends to a client to be processed.

rule: A condition or action, or a set of conditions or actions, that performs tasks automatically based on events and values.

rules table: A **Table object** whose rows represent the rules that are contained in a **Folder object**.

search criteria: A criteria used to determine which messages are included in a folder with specific characteristics. It is composed of a restriction, which is the filter to be applied, and a search scope, which are the folders that contain the content to search.

sort order: A set of rules in a search query that defines the ordering of rows in the search result. Each rule consists of a managed property, such as modified date or size, and a direction for order, such as ascending or descending. Multiple rules are applied sequentially.

Table object: An object that is used to view properties for a collection of objects of a specific type, such as a Message object or a **Folder object**. A Table object is structured in a row and column format with each row representing an object and each column representing a property of the object.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

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

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

[MS-OXCMAPIHTTP] Microsoft Corporation, "Messaging Application Programming Interface (MAPI) Extensions for HTTP".

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

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

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

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

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

1.2.2 Informative References

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

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

[MS-OXCPERM] Microsoft Corporation, "Exchange Access and Operation Permissions Protocol".

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

[MS-OXOMSG] Microsoft Corporation, "Email Object Protocol".

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

1.3 Overview

The Table Object Protocol is used to read tabular data from a server. It specifies a set of operations that a client can use to request tabular data from a server based on a **handle** to the table as described in section 4.4. The client can specify the columns, the **restriction**, and the **sort order** for the table, and can request that the rows of the table be categorized according to specific properties as described in section 4.5. The client can then request one or more rows of data. Additionally, the client can find rows, navigate through the rows, and create bookmarks for easier navigation. The protocol can provide a way to free server resources associated with bookmarks.

When the client requests that the rows of the table be categorized, the server will include **header rows** in the table that don't have the same properties as normal rows. The client can request that the server hide or show all of the normal rows for which the header row represents their **category**. A

category can be nested inside another category. The client can retrieve a **binary large object (BLOB)** that specifies which categories are collapsed and which are expanded in the current table. This BLOB can then be given to the table at a future time to restore the collapsed state of the table as well as the cursor location.

Multivalue instances can be retrieved from the table when a **multivalue property** is specified in the **column set**. When multivalue instances are requested, for each value in a multivalue property there will be an **instance** of the row that has that single value for the property. All other properties are repeated in each multivalue instance.

Categories that are based on multivalue properties will display the multivalue instances under each header row representing a value that is set on that row. The row that is displayed under a given header row will include the single property value specified by the header row, not all values for the property.

Some tables might not support certain table operations. For example, **rules tables** do not support sorting and return an error if sorting is attempted. Tables that do not support asynchronous operations can perform them synchronously or return an error.

1.3.1 Table Notifications

Tables are not static representations of the data. Table rows can be modified, moved, created, and deleted while the client is using the table. Table notifications are used to inform the client of all changes made to the table since it was opened.

To properly use the Table Object Protocol, both client and server need to implement the Core Notifications Protocol, as described in [MS-OXCNOTIF].

1.4 Relationship to Other Protocols

The Table Object Protocol uses the Remote Operations (ROP) List and Encoding Protocol, as described in [MS-OXCROPS], and the Core Notifications Protocol, as described in [MS-OXCNOTIF].

The Message and Attachment Object Protocol, the Email Rules Protocol, the Exchange Access and Operation Permissions Protocol, and the Folder Object Protocol depend on the Table Object Protocol. For information about these protocols, see [MS-OXCMSG], [MS-OXORULE], [MS-OXCPERM], and [MS-OXCFOLD], respectively.

For conceptual background information and overviews of the relationships and interactions between this and other protocols, see [MS-OXPROTO].

1.5 Prerequisites/Preconditions

The Table Object Protocol assumes that the client has acquired a handle to the **Table object** on which it is going to operate. The method by which a handle to a Table object is acquired is dependent on the table type. For information about how to obtain a handle to a specific table type, see the appropriate document, as stated in the following list:

- **Contents table** - [MS-OXCFOLD] section 2.2.1.14
- **Hierarchy table** - [MS-OXCFOLD] section 2.2.1.13
- **Attachments table** - [MS-OXCMSG] section 2.2.3.17
- **Permissions table** - [MS-OXCPERM] section 2.2.1
- Rules table - [MS-OXORULE] section 2.2.2

1.6 Applicability Statement

The Table Object Protocol is used to query tabular data associated with folders, messages, attachments, **permissions**, and **rules** on a server.

1.7 Versioning and Capability Negotiation

The Table Object Protocol does not support negotiation of the version to be used. Instead, the client determines the version of the server to which it has connected. The client's behavior is limited by the capabilities of the server version with which it communicates.

To get information about version limitations, the client checks the version number that is returned by the server in either the **EcDoConnectEx** method, as described in [MS-OXCRPC], or the **X-ServerApplication** header of the **Connect** request type response, as described in [MS-OXCMAPIHTTP].

A feature, packed buffers for the **RopQueryRows** **remote operation (ROP)** ([MS-OXCROPS] section 2.2.5.4), is available for servers with a major version of at least eight (8), as described in [MS-OXCRPC] section 3.1.4.1.3.2.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

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

2.2 Message Syntax

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

2.2.1 Table-Specific Properties

The following properties can be included in the column set of a table for the purpose of collapsing and expanding categories. The following properties are used by the client but are produced by the server. For more details about these properties, see [MS-OXPROPS].

2.2.1.1 PidTagInstID

Data type: **PtypInteger64** ([MS-OXCDATA] section 2.11.1)

The **PidTagInstID** property ([MS-OXPROPS] section 2.736) is an identifier for all instances of a row in the table. When a **RopGetCollapseState** **ROP request** ([MS-OXCROPS] section 2.2.5.18) is sent, the client passes this property value in the **RowId** field, as specified in [MS-OXCROPS] section 2.2.5.18.1, to specify a cursor to store. When the **PidTagInstID** property is included in the column set, the server sets the property to the same value for each row that is an instance of the same underlying data.

2.2.1.2 PidTagInstanceNum

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

The **PidTagInstanceNum** property ([MS-OXPROPS] section 2.735) is an identifier for a single instance of a row in the table. When a **RopGetCollapseState** ROP request ([MS-OXCROPS] section 2.2.5.18) is sent, the client passes this property value in the **RowInstanceNumber** field to specify a cursor to store. When this property is included in the column set, the server sets it to a different value for each row that is an instance of the same underlying data.

2.2.1.3 PidTagRowType

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

The **PidTagRowType** property ([MS-OXPROPS] section 2.931) identifies the type of the row. The possible values are given in the following table.

Row name	Value	Meaning
TBL_LEAF_ROW	0x00000001	The row is a row of data.
TBL_EMPTY_CATEGORY	0x00000002	The row is a header row with no rows inside the category.
TBL_EXPANDED_CATEGORY	0x00000003	The row is a header row that is expanded.
TBL_COLLAPSED_CATEGORY	0x00000004	The row is a header row that is collapsed.

2.2.1.4 PidTagDepth

Data type: **PtypInteger32** property ([MS-OXCDATA] section 2.11.1)

The **PidTagDepth** property ([MS-OXPROPS] section 2.664) specifies the number of nested categories in which a given row is contained. For example, if a row is contained within two header rows, its depth is 0x0002. When a table contains no categories, all rows will have a depth of 0x0000.

2.2.1.5 PidTagContentCount

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

The **PidTagContentCount** property ([MS-OXCFOLD] section 2.2.2.2.1.1) specifies the number of rows under the header row. This property is set whether the header row is collapsed or expanded.

2.2.1.6 PidTagContentUnreadCount

Data type: **PtypInteger32** property ([MS-OXCDATA] section 2.11.1)

The **PidTagContentUnreadCount** property ([MS-OXCFOLD] section 2.2.2.2.1.2) specifies the number of rows under the header row that have the **PidTagRead** property ([MS-OXPROPS] section 2.869) set to "FALSE" (0x00). This value is set whether the header row is collapsed or expanded.

2.2.2 Table ROPs

The following sections specify the semantics of ROP fields that are specific to the Table Object Protocol. Before sending these requests to the server, the handle to the Table object that is used in the ROP requests MUST be acquired.

2.2.2.1 Table ROP Constants

2.2.2.1.1 Predefined Bookmarks

The following values are used in the **RopSeekRow** ROP request ([MS-OXCROPS] section 2.2.5.8), the **RopQueryRows** ROP response ([MS-OXCROPS] section 2.2.5.4), and the **RopFindRow** ROP request ([MS-OXCROPS] section 2.2.5.13).

Constant name	Value	Meaning
BOOKMARK_BEGINNING	0x00	Points to the beginning position of the table, or the first row.
BOOKMARK_CURRENT	0x01	Points to the current position of the table, or the current row.
BOOKMARK_END	0x02	Points to the ending position of the table, or the location after the last row.

2.2.2.1.2 Custom Bookmarks

The following value is used in the **RopFindRow** ROP request ([MS-OXCROPS] section 2.2.5.13).

Constant name	Value	Meaning
BOOKMARK_CUSTOM	0x03	Points to the custom position in the table. Used with the **BookmarkSize** and

Constant name	Value	Meaning
		Bookmark fields.

2.2.2.1.3 TableStatus

The **TableStatus** field refers to the status of any asynchronous operations being performed on the table.<1> The following values are used in the **RopGetStatus** ROP ([MS-OXCROPS] section 2.2.5.6), **RopAbort** ROP ([MS-OXCROPS] section 2.2.5.5), **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1), **RopRestrict** ROP **(**[MS-OXCROPS] section 2.2.5.3), and **RopSortTable** ROP ([MS-OXCROPS] section 2.2.5.2) responses.

Constant name	Value	Meaning
TBLSTAT_COMPLETE	0x00	No operations are in progress.
TBLSTAT_SORTING	0x09	A **RopSortTable** ROP is in progress.
TBLSTAT_SORT_ERROR	0x0A	An error occurred during a **RopSortTable** ROP.
TBLSTAT_SETTING_COLS	0x0B	A **RopSetColumns** ROP is in progress.
TBLSTAT_SETCOL_ERROR	0x0D	An error occurred during a **RopSetColumns** ROP.
TBLSTAT_RESTRICTING	0x0E	A **RopRestrict** ROP is in progress.
TBLSTAT_RESTRICT_ERROR	0x0F	An error occurred during a **RopRestrict** ROP.

2.2.2.1.4 Asynchronous Flags

The asynchronous **flags** specify whether certain ROPs are to be performed asynchronously.<2>

Flag name	Value	Meaning
TBL_ASYNC	0x01	The server SHOULD perform the ROP asynchronously. For details about asynchronous table operations, see sections 3.1.4.1.1 and 3.2.5.1. If the server does not honor requests to perform operations asynchronously, asynchronous flags will be ignored by the server (they are not read or written to).
	0x00	The server will perform the operation synchronously.

2.2.2.2 RopSetColumns ROP

The **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1) sets the properties that the client requests to be included in the table. This ROP is valid only on Table objects.

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

2.2.2.2.1 RopSetColumns ROP Request Buffer

The following descriptions define valid fields for the **RopSetColumns** ROP request buffer ([MS-OXCROPS] section 2.2.5.1.1).

SetColumnsFlags (1 byte): A structure that contains an OR'ed combination of the asynchronous flags, which are specified in section 2.2.2.1.4.<3> This field MUST NOT have any of the other bits set.

PropertyTagCount (2 bytes): An unsigned integer that specifies the number of **property tags** in the **PropertyTags** field. This value MUST be 1 or greater.

PropertyTags (variable): An array of **PropertyTag** structures, as described in [MS-OXCDATA] section 2.9, that identify the set and order of property values to be returned by the server in the ROP response buffer of the **RopQueryRows** ([MS-OXCROPS] section 2.2.5.4), **RopFindRow** ([MS-OXCROPS] section 2.2.5.13), and **RopExpandRow** ([MS-OXCROPS] section 2.2.5.16) ROPs, as specified in sections 2.2.2.5, 2.2.2.14, and 2.2.2.17, respectively.

Every table MUST have at least one column. If the **property type** is a multivalue property, and the client wants multivalue instances based on this property, it MUST also set the **MultivalueInstance** bit (0x2000), as specified in [MS-OXCDATA] section 2.11.1.3, of the **PropertyTag** structure. If the property type is not multivalued, it does not set the **MultivalueInstance** bit of the **PropertyTag** structure set

2.2.2.2.2 RopSetColumns ROP Response Buffer

The following description defines a valid field for the **RopSetColumns** ROP response buffer ([MS-OXCROPS] section 2.2.5.1.2).

TableStatus (1 byte): An enumeration that indicates the status of asynchronous operations being performed on the table.<4> It MUST have one of the table status values that are specified in section 2.2.2.1.3.

2.2.2.3 RopSortTable ROP

The **RopSortTable** ROP ([MS-OXCROPS] section 2.2.5.2) orders the rows of a contents table based on sort criteria. This ROP is valid only on Table objects.

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

2.2.2.3.1 RopSortTable ROP Request Buffer

The following descriptions define valid fields for the **RopSortTable** ROP request buffer ([MS-OXCROPS] section 2.2.5.2.1).

SortTableFlags (1 byte): A structure that contains an OR'ed combination of the asynchronous flags, which are specified in section 2.2.2.1.4.<5> This field MUST NOT have any of the other bits set.

SortOrderCount (2 bytes): An unsigned integer that specifies the number of **SortOrder** structures, as specified in [MS-OXCDATA] section 2.13.1, in the **SortOrders** field.

CategoryCount (2 bytes): An unsigned integer that specifies the number of **SortOrder** structures, as specified in [MS-OXCDATA] section 2.13.1, in the **SortOrders** field, that are designated as category columns. The **SortOrder** structures occupy the first **CategoryCount** field positions in the **SortOrders** array. The value of the **CategoryCount** field MUST be in the range 0 to the value of the **SortOrderCount** field.

ExpandedCount (2 bytes): An unsigned integer that specifies the number of categories that start in the expanded state. This value MUST be in the range 0 to the value of the **CategoryCount** field. The first categories of the **ExpandedCount** field are initially expanded. If the value of the

CategoryCount field is equal to the value of the **ExpandedCount** field, then all categories are expanded.

SortOrders (variable): An array of **SortOrder** structures, as specified in [MS-OXCDATA] section 2.13.1, that defines the sort to be performed. The number of **SortOrder** structures in the array MUST be equal to the value of the **SortOrderCount** field. For categorized sorting, the **SortOrder** structure specifies the property type and **property ID** that are used as the category. When the value of the **SortOrderCount** field exceeds the value of the **CategoryCount** field, indicating that there are more sort keys than categories, categories are created from the **SortOrder** structures that appear first in the **SortOrders** array. The remaining **SortOrder** structures are used to sort the rows within the categories.

For example, if the **SortOrderCount** field is set to 0x0003 and the **CategoryCount** field is set to 0x0002, the columns described by the **PropertyType** and **PropertyId** members of the first two entries in the **SortOrders** field are used as the category columns. The first entry serves as the top-level category grouping; the second entry as the secondary grouping. All of the rows that match the two category columns are sorted using the sort key defined in the third entry.

If a **SortOrder** structure specifies a multivalue property, it MUST also have the **MultivalueInstance** bit set, as specified in [MS-OXCDATA] section 2.11.1.3, specifying that the sort be performed using the individual values of that property. Sort order on a multivalue property that is not also being used for multivalue instances is undefined. The **SortOrders** field MUST NOT contain more than one **SortOrder** structure specifying a multivalue property in the first **CategoryCount** field positions of the array.

If the **Order** member of a **SortOrder** structure is set to "Ascending", the table will be sorted in ascending order by the column specified in the **PropertyType** and **PropertyId** members.

If the **Order** member of a **SortOrder** structure is set to "Descending", the table will be sorted in descending order by the column specified in the **PropertyType** and **PropertyId** members.

If the **Order** member of a **SortOrder** structure is set to "MaximumCategory", that structure directly follows the first **CategoryCount** structure in the **SortOrders** field and the value of the **CategoryCount** field MUST be greater than 0x0000 (zero). This **SortOrder** structure, at position **CategoryCount** + 1 in the array, modifies the immediately previous category sort. The categories in the table will be not be sorted by the column specified in the category sort. The categories will be sorted according to each category's maximum value of the specified column. Any **SortOrder** structures after this one will subsort the rows within each category.

2.2.2.3.2 RopSortTable ROP Response Buffer

The following description defines a valid field for the **RopSortTable** ROP response buffer ([MS-OXCROPS] section 2.2.5.2.2).

TableStatus (1 byte): An enumeration that indicates the status of asynchronous operations being performed on the table.<6> This field MUST have one of the table status values that are specified in section 2.2.2.1.3.

2.2.2.4 RopRestrict ROP

The **RopRestrict** ROP ([MS-OXCROPS] section 2.2.5.3) establishes a restriction on a table. Applying a restriction has no effect on the underlying data of a table; it simply alters the table by limiting the rows that can be retrieved to rows containing data that satisfy the restriction. This ROP is valid only on Table objects.

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

2.2.2.4.1 RopRestrict ROP Request Buffer

The following descriptions define valid fields for the **RopRestrict** ROP request buffer ([MS-OXCROPS] section 2.2.5.3.1).

RestrictFlags (1 byte): A structure that contains an OR'ed combination of the asynchronous flags, as specified in section 2.2.2.1.4.<7> This field MUST NOT have any of the other bits set.

RestrictionDataSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **RestrictionData** field.

RestrictionData (variable): A restriction that is applied to the table. For details about restrictions, see [MS-OXCDATA] section 2.12. This field has a number of bytes equal to the value of the **RestrictionDataSize** field.

2.2.2.4.2 RopRestrict ROP Response Buffer

The following description defines a valid field for the **RopRestrict** ROP response buffer ([MS-OXCROPS] section 2.2.5.3.2).

TableStatus (1 byte): An enumeration that indicates the status of asynchronous operations being performed on the table.<8> This field MUST have one of the table status values that are specified in section 2.2.2.1.3.

2.2.2.5 RopQueryRows ROP

The **RopQueryRows** ROP ([MS-OXCROPS] section 2.2.5.4) returns zero or more rows from a table, beginning from the current table cursor position. This ROP is valid only on Table objects.

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

2.2.2.5.1 RopQueryRows ROP Request Buffer

The following descriptions define valid fields for the **RopQueryRows** ROP request buffer ([MS-OXCROPS] section 2.2.5.4.1).

QueryRowsFlags (1 byte): A structure that contains any of the following bit flags.

Flag name	Value	Meaning
Advance	0x00	Advance the table cursor.
NoAdvance	0x01	Do not advance the table cursor.
EnablePackedBuffers	0x02	Enable packed buffers for the response. To allow packed buffers to be used, this flag is used in conjunction with the **Chain** flag (0x00000004) that is passed in either the *pulFlags* parameter of the **EcDoRpcExt2** method, as specified in [MS-OXCRPC] section 3.1.4.2, or the **Flags** field of the **Execute** request type request body,<9> as specified in [MS-OXCMAPIHTTP] section 2.2.4.2.1. For details about extended-buffer packing, see [MS-OXCRPC]. This flag is only supported against servers with a major version of at least eight (8), as described in section 1.7.

The **QueryRowsFlags** field MUST NOT have both the **Advance** and **NoAdvance** flags set simultaneously. The field MUST NOT have any of the other bits set.

ForwardRead (1 byte): The direction in which to retrieve rows. It is set to "TRUE" (0x01) to read the table forwards. It is set to "FALSE" (0x00) to read the table backwards. It MUST NOT be set to any other value.

RowCount (2 bytes): The maximum number of rows to be returned

2.2.2.5.2 RopQueryRows ROP Response Buffer

The following descriptions define valid fields for the **RopQueryRows** ROP response buffer ([MS-OXCROPS] section 2.2.5.4.2).

Origin (1 byte): An enumeration that identifies the cursor position. This field MUST be set to one of the predefined **bookmark** values specified in section 2.2.2.1.1.

RowCount (2 bytes): An unsigned integer that specifies the number of rows returned. Its value MUST be less than or equal to the **RowCount** field value that is specified in the request, and it MUST be greater than or equal to 0x0000. It MUST be equal to the number of **PropertyRow** objects returned in the **RowData** field.

RowData (variable): A list of **PropertyRow** structures that contains the array of rows returned. Each row is represented by a **PropertyRow** object, as specified in [MS-OXCDATA] section 2.8.1. Each row MUST have the same columns and ordering of columns as specified in the last **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1).

The **RowData** field MUST NOT include rows that don't match the criteria specified in the last **RopRestrict** ROP request ([MS-OXCROPS] section 2.2.5.3). If the **RopRestrict** ROP has not been issued, the **RowData** field MUST include all rows.

The rows MUST be sorted and grouped according to the sort order specified in the last **RopSortTable** ROP request ([MS-OXCROPS] section 2.2.5.2). If the **RopSortTable** ROP has not been sent, the default sort order is undefined.

Every property value returned in a row MUST be less than or equal to 510 bytes in size. If a property value is greater than 510 bytes in size, it MUST be truncated to 510 bytes.

2.2.2.6 RopAbort ROP

The **RopAbort** ROP ([MS-OXCROPS] section 2.2.5.5) attempts to stop any asynchronous table operations that are currently in progress.<10> This ROP is valid only on Table objects.

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

2.2.2.6.1 RopAbort ROP Request Buffer

The following descriptions define valid fields for the **RopAbort** ROP request buffer ([MS-OXCROPS] section 2.2.5.5.1).

This protocol adds no additional field information to the **RopAbort** ROP request buffer.

2.2.2.6.2 RopAbort ROP Response Buffer

The following description defines a valid field for the **RopAbort** ROP response buffer ([MS-OXCROPS] section 2.2.5.5.2).

TableStatus (1 byte): An enumeration that indicates the status of asynchronous operations being performed on the table before the abort.<11> Its value MUST be one of the table status values that are specified in section 2.2.2.1.3.

2.2.2.7 RopGetStatus ROP

The **RopGetStatus** ROP ([MS-OXCROPS] section 2.2.5.6) retrieves information about the current status of asynchronous operations being performed on the table.<12> This ROP is valid only on Table objects.

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

2.2.2.7.1 RopGetStatus ROP Request Buffer

The following descriptions define valid fields for the **RopGetStatus** ROP request buffer ([MS-OXCROPS] section 2.2.5.6.1).

This protocol adds no additional field information to the **RopGetStatus** ROP request buffer.

2.2.2.7.2 RopGetStatus ROP Response Buffer

The following description defines a valid field for the **RopGetStatus** ROP response buffer ([MS-OXCROPS] section 2.2.5.6.2).

TableStatus (1 byte): An enumeration that indicates the status of asynchronous operations being performed on the table.<13> Its value MUST be one of the table status values that are specified in section 2.2.2.1.3.

2.2.2.8 RopQueryPosition ROP

The **RopQueryPosition** ROP ([MS-OXCROPS] section 2.2.5.7) returns the location of the cursor in the table. Note that the current position and total number of rows could change based on external events before the response to this message is received. This ROP is valid only on Table objects.

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

2.2.2.8.1 RopQueryPosition ROP Request Buffer

The following descriptions define valid fields for the **RopQueryPosition** ROP request buffer ([MS-OXCROPS] section 2.2.5.7.1).

This protocol adds no additional field information to the **RopQueryPosition** ROP request buffer.

2.2.2.8.2 RopQueryPosition ROP Response Buffer

The following descriptions define valid fields for the **RopQueryPosition** ROP response buffer ([MS-OXCROPS] section 2.2.5.7.2).

Numerator (4 bytes): An unsigned integer that indicates the index (0-based) of the current row. Its value MUST be greater than or equal to 0x00000000.

Denominator (4 bytes): An unsigned integer that indicates the total number of rows in the table. Its value MUST be greater than or equal to the value of the **Numerator** field.

2.2.2.9 RopSeekRow ROP

The **RopSeekRow** ROP ([MS-OXCROPS] section 2.2.5.8) moves the table cursor to a specific location in the table. The new location is specified by a predefined bookmark, as specified in section 2.2.2.1.1,

and the number of rows to move (forward or backwards) from that bookmark. This ROP is valid only on Table objects.

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

2.2.2.9.1 RopSeekRow ROP Request Buffer

The following descriptions define valid fields for the **RopSeekRow** ROP request buffer ([MS-OXCROPS] section 2.2.5.8.1).

Origin (1 byte): An enumeration that contains the **Bookmark** field array indicating the starting position of the seek operation. This field's value MUST be one of the predefined bookmark values that are specified in section 2.2.2.1.1.

RowCount (4 bytes): A signed integer that indicates the number of rows to seek, starting from the bookmark. To seek forward from the bookmark, the value MUST be positive; to seek backwards, the value MUST be negative.

WantRowMovedCount (1 byte): A Boolean that specifies whether the actual number of rows moved MUST be returned by the server. This field MUST be set to "TRUE" (0x01) or "FALSE" (0x00). If this field is set to "TRUE" (0x01), the server MUST return the actual number of rows moved. If this field is set to "FALSE" (0x00), the value of the **HasSoughtLess** and **RowsSought** fields specified in section 2.2.2.9.2 are undefined, and the client MUST ignore them.

The actual number of rows moved can differ from the requested number of rows if the beginning or end of the table is encountered before moving the requested number of rows.

2.2.2.9.2 RopSeekRow ROP Response Buffer

The following descriptions define valid fields for the **RopSeekRow** ROP response buffer ([MS-OXCROPS] section 2.2.5.8.2).

HasSoughtLess (1 byte): A Boolean that specifies whether the number of rows moved is less than the number of rows requested. This field MUST be set to "TRUE" (0x01) if the number of rows moved is less than the number of rows requested (**RowCount** field), otherwise it MUST be set to "FALSE" (0x00).

The **HasSoughtLess** field MUST be present in the response. This field's value MUST be valid if the **WantRowMovedCount** field (in the request) is set to "TRUE", and its value MUST be ignored if **WantRowMovedCount** is set to "FALSE".

RowsSought (4 bytes): A signed integer that specifies the actual number of rows moved. If the value of the **RowCount** field (in the request) is negative, the value of the **RowsSought** field MUST also be negative or 0x00000000, indicating that the seek was performed backwards. If the value of the **RowCount** field (in the request) is positive, then the value of the **RowsSought** field MUST also be positive.

This field MUST be present in the response. This field's value MUST be valid if the **WantRowMovedCount** field (in the request) is set to "TRUE" and MUST be ignored if the **WantRowMovedCount** field is set to "FALSE".

2.2.2.10 RopSeekRowBookmark ROP

The **RopSeekRowBookmark** ROP ([MS-OXCROPS] section 2.2.5.9) moves the table cursor to a specific location in the table. The new location is specified by a custom bookmark, as specified in section 2.2.2.1.2, and the number of rows to move (forward or backwards) from that bookmark.

This ROP is distinguished from the **RopSeekRow** ROP ([MS-OXCROPS] section 2.2.5.8) in that the bookmark is not a predefined one, as specified in section 2.2.2.1.1, but one created by a **RopCreateBookmark** ROP request ([MS-OXCROPS] section 2.2.5.11). This ROP is valid only on Table objects.

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

2.2.2.10.1 RopSeekRowBookmark ROP Request Buffer

The following descriptions define valid fields for the **RopSeekRowBookmark** ROP request buffer ([MS-OXCROPS] section 2.2.5.9.1).

BookmarkSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **Bookmark** field.

Bookmark (variable): A bookmark indicating the starting position of the seek operation. The value of the **Bookmark** field MUST be data that was returned by a previous **RopCreateBookmark** ROP request ([MS-OXCROPS] section 2.2.5.11). The bookmark MUST NOT have been previously freed by using the **RopFreeBookmark** ROP ([MS-OXCROPS] section 2.2.5.14).

RowCount (4 bytes): A signed integer that specifies the number of rows to seek, starting from the bookmark. To seek forward from the bookmark, the value MUST be positive; to seek backward, the value MUST be negative.

WantRowMovedCount (1 byte): A Boolean that specifies whether the actual number of rows moved is returned by the server. If this field is set to "TRUE" (0x01), the server MUST return the actual number of rows moved. If this field is set to "FALSE" (0x00), the value of the **HasSoughtLess** and **RowsSought** fields specified in section 2.2.2.10.2 is undefined, and the client MUST ignore them.

The actual number of rows moved can differ from the requested number of rows if the beginning or end of the table is encountered before moving the requested number of rows.

2.2.2.10.2 RopSeekRowBookmark ROP Response Buffer

The following descriptions define valid fields for the **RopSeekRowBookmark** ROP response buffer ([MS-OXCROPS] section 2.2.5.9.2).

RowNoLongerVisible (1 byte): A Boolean that indicates whether the row to which the bookmark pointed is no longer visible. This field MUST be set to "TRUE" (0x01) if the row to which the bookmark pointed has been removed from the table. (For example, the row's properties changed so that they didn't match the restriction, the row was deleted, or the row's header row has been collapsed.) Otherwise, this field MUST be set to "FALSE" (0x00).

When the row to which the bookmark pointed is no longer visible, the bookmark will point to the next row in the table. In this case, the seek will begin from the next row after the bookmark in the table.

HasSoughtLess (1 byte): A Boolean that specifies whether the number of rows moved is less than the number of rows requested. This field MUST be set to "TRUE" (0x01) if the number of rows moved is less than the number of rows requested in the **RowCount** field; otherwise it MUST be set to "FALSE" (0x00).

The **HasSoughtLess** field MUST be present in the response. This field's value MUST be valid if the **WantRowMovedCount** field (in the request) is set to "TRUE" (0x01), and its value MUST be ignored if **WantRowMovedCount** is set to "FALSE".

RowsSought (4 bytes): An unsigned integer that specifies the actual number of rows moved. If the value of the **RowCount** field in the request is negative, the value of the **RowsSought** field MUST also

be negative or zero (0x00000000), indicating that the seek was performed backwards. If the value of the **RowCount** field in the request is positive, the value of **RowsSought** MUST also be positive.

2.2.2.11 RopSeekRowFractional ROP

The **RopSeekRowFractional** ROP ([MS-OXCROPS] section 2.2.5.10) moves the table cursor to an approximate position in the table.<14> The new location is specified as a fraction of the table size. This ROP is valid only on Table objects.

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

2.2.2.11.1 RopSeekRowFractional ROP Request Buffer

The following descriptions define valid fields for the **RopSeekRowFractional** ROP request buffer ([MS-OXCROPS] section 2.2.5.10.1).

Numerator (4 bytes): The numerator of the fractional position. The value SHOULD be less than or equal to the value of the **Denominator** field.

If the value of **Numerator** field is zero (0x00000000), the cursor MUST be set to the first row in the table. If the value is equal to or greater than the value of **Denominator** field, the cursor MUST be set past the last row in the table.

Denominator (4 bytes): The denominator of the fractional position. The value MUST NOT be set to zero (0x00000000).

2.2.2.11.2 RopSeekRowFractional ROP Response Buffer

The following descriptions define valid fields for the **RopSeekRowFractional** ROP response buffer ([MS-OXCROPS] section 2.2.5.10.2).

This protocol adds no additional field information to the **RopSeekRowFractional** ROP response buffer.

2.2.2.12 RopCreateBookmark ROP

The **RopCreateBookmark** ROP ([MS-OXCROPS] section 2.2.5.11) creates a new bookmark at the current cursor position in the table. This ROP is valid only on Table objects.

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

2.2.2.12.1 RopCreateBookmark ROP Request Buffer

The following descriptions define valid fields for the **RopCreateBookmark** ROP request buffer ([MS-OXCROPS] section 2.2.5.11.1).

This protocol adds no additional field information to the **RopCreateBookmark** ROP request buffer.

2.2.2.12.2 RopCreateBookmark ROP Response Buffer

The following descriptions define valid fields for the **RopCreateBookmark** ROP response buffer ([MS-OXCROPS] section 2.2.5.11.2).

BookmarkSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **Bookmark** field. This field MUST be present.

Bookmark (variable): An array of bytes that contains the bookmark data. This data is specific to the server. The client MUST NOT assume that this data has a specific format.

2.2.2.13 RopQueryColumnsAll ROP

The **RopQueryColumnsAll** ROP ([MS-OXCROPS] section 2.2.5.12) returns a complete list of all columns for the table. The list includes all columns that the server has for the table, not necessarily only those requested by a **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1). This ROP is valid only on Table objects.

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

2.2.2.13.1 RopQueryColumnsAll ROP Request Buffer

The following descriptions define valid fields for the **RopQueryColumnsAll** ROP request buffer ([MS-OXCROPS] section 2.2.5.12.1).

This protocol adds no additional field information to the **RopQueryColumns** ROP request buffer.

2.2.2.13.2 RopQueryColumnsAll ROP Response Buffer

The following descriptions define valid fields for the **RopQueryColumnsAll** ROP response buffer ([MS-OXCROPS] section 2.2.5.12.2).

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

PropertyTags (variable): An array of property tags, each of which corresponds to an available column in the table. Each property tag is represented by a **PropertyTag** structure, as specified in [MS-OXCDATA] section 2.9.

2.2.2.14 RopFindRow ROP

The **RopFindRow** ROP ([MS-OXCROPS] section 2.2.5.13) returns the next row in a table that matches the **search criteria** and moves the cursor to that row. The initial location for the search is specified by a bookmark. This ROP is valid only on Table objects.

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

2.2.2.14.1 RopFindRow ROP Request Buffer

The following descriptions define valid fields for the **RopFindRow** ROP request buffer ([MS-OXCROPS] section 2.2.5.13.1).

FindRowFlags (1 byte): A structure that contains an OR'ed combination of any of the following flags.

Value	Meaning
0x00	Perform the find forwards.

Value	Meaning
0x01	Perform the find backwards.

This field MUST NOT have any of the other bits set.

RestrictionDataSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **RestrictionData** field.

RestrictionData (variable): A restriction packet that contains a restriction that specifies the criteria to be used for the search. Restrictions are further specified in [MS-OXCDATA] section 2.12.

2.2.2.14.2 RopFindRow ROP Response Buffer

The following descriptions define valid fields for the **RopFindRow** ROP response buffer ([MS-OXCROPS] section 2.2.5.13.2).

RowNoLongerVisible (1 byte): A Boolean that indicates whether the row to which the bookmark pointed is no longer visible. This field SHOULD<15> be set to "TRUE" (0x01) if the row to which the bookmark pointed has been removed from the table. (For example, the row's properties changed so that it didn't match the restriction, the row was deleted, or the row's header row has been collapsed.) Otherwise, this field MUST be set to "FALSE" (0x00).

When the row to which the bookmark pointed is no longer visible, the search will begin from the next row after the bookmark in the table. When searching backward, the search will not consider the row to which the bookmark is currently pointing, but will begin at the previous row.

HasRowData (1 byte): A Boolean that specifies whether a row is included in this response. If a row that meets the specified search criteria was found and the row data is included in the response, this field MUST be set to "TRUE" (0x01).

RowData (variable): A **PropertyRow** structure, as specified in [MS-OXCDATA] section 2.8.1, that specifies the row. If the value of the **HasRowData** field is "TRUE" (0x01), the **RowData** field MUST be present. If the value of **HasRowData** is "FALSE" (0x00), the **RowData** field MUST NOT be present.

2.2.2.15 RopFreeBookmark ROP

The **RopFreeBookmark** ROP ([MS-OXCROPS] section 2.2.5.14) frees the memory associated with a bookmark that was returned by a previous **RopCreateBookmark** ROP request ([MS-OXCROPS] section 2.2.5.11). If the bookmark is released by the **RopFreeBookmark** ROP, the server will return the ecInvalidBookmark error code when it is used.. This ROP is valid only on Table objects. If the bookmark has been released by the **ROPRelease** ROP ([MS-OXCROPS] section 2.2.15.3), attempts to use the bookmark will fail with the ecNullObject error code.

The server processing behaviors for the **RopFreeBookmark** ROP, including the complete list of possible error codes returned, are further specified in section 3.2.5.15.

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

2.2.2.15.1 RopFreeBookmark ROP Request Buffer

The following descriptions define valid fields for the **RopFreeBookmark** ROP request buffer ([MS-OXCROPS] section 2.2.5.14.1).

BookmarkSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **Bookmark** field.

Bookmark (variable): An array that specifies the bookmark to be freed. The bookmark MUST be one that was returned by a previous **RopCreateBookmark** ROP request ([MS-OXCROPS] section 2.2.5.11). The bookmark MUST NOT have been previously freed using the **RopFreeBookmark** ROP.

2.2.2.15.2 RopFreeBookmark ROP Response Buffer

The following descriptions define valid fields for the **RopFreeBookmark** ROP response buffer ([MS-OXCROPS] section 2.2.5.14.2).

This protocol adds no additional field information to the **RopFreeBookmark** ROP response buffer.

2.2.2.16 RopResetTable ROP

The **RopResetTable** ROP ([MS-OXCROPS] section 2.2.5.15) performs the following actions:

- Removes the existing column set, restriction, and sort order from the table.
- Invalidates bookmarks.
- Resets the cursor to the beginning of the table.

This ROP is valid only on Table objects.

After sending this ROP, a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) MUST be sent before sending a **RopFindRow** ([MS-OXCROPS] section 2.2.5.13), **RopQueryRows** ([MS-OXCROPS] section 2.2.5.4), or **RopExpandRow** ROP request ([MS-OXCROPS] section 2.2.5.16). Existing bookmarks SHOULD be freed using the **RopFreeBookmark** ROP ([MS-OXCROPS] section 2.2.5.14). The client can choose not to send a **RopFreeBookmark** ROP; however, this can degrade server performance until the table is released by using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

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

2.2.2.16.1 RopResetTable ROP Request Buffer

The following descriptions define valid fields for the **RopResetTable** ROP request buffer ([MS-OXCROPS] section 2.2.5.15.1).

This protocol adds no additional field information to the **RopResetTable** ROP request buffer.

2.2.2.16.2 RopResetTable ROP Response Buffer

The following descriptions define valid fields for the **RopResetTable** ROP response buffer ([MS-OXCROPS] section 2.2.5.15.2).

This protocol adds no additional field information to the **RopResetTable** ROP response buffer.

2.2.2.17 RopExpandRow ROP

The **RopExpandRow** ROP ([MS-OXCROPS] section 2.2.5.16) expands a collapsed category of a table and returns the rows that belong in the newly expanded category. The maximum number of **leaf rows** to be returned can be specified. This ROP is valid only on Table objects.

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

2.2.2.17.1 RopExpandRow ROP Request Buffer

The following descriptions define valid fields for the **RopExpandRow** ROP request buffer ([MS-OXCROPS] section 2.2.5.16.1).

MaxRowCount (2 bytes): An unsigned integer that specifies the maximum number of leaf rows to be returned in the response. This field SHOULD<16> be set to a value of 0.

CategoryId (8 bytes): An identifier that specifies the category to be expanded. This field is set to the value of the **PidTagInstID** property (section 2.2.1.1) of the category's header row.

2.2.2.17.2 RopExpandRow ROP Response Buffer

The following descriptions define valid fields for the **RopExpandRow** ROP response buffer ([MS-OXCROPS] section 2.2.5.16.2).

ExpandedRowCount (4 bytes): An unsigned integer that specifies the total number of rows that are in the expanded category.

RowCount (2 bytes): An unsigned integer that specifies the number of **PropertyRow** structures, as specified in [MS-OXCDATA] section 2.8.1, that are contained in the **RowData** field. The value of this field MUST be less than or equal to both of the following:

- The value of the **MaxRowCount** field in the ROP request buffer; and
- The value of the **ExpandedRowCount** field in the ROP response buffer.

2.2.2.18 RopCollapseRow ROP

The **RopCollapseRow** ROP ([MS-OXCROPS] section 2.2.5.17) collapses an expanded category. This ROP is valid only on Table objects.

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

2.2.2.18.1 RopCollapseRow ROP Request Buffer

The following description defines a valid field for the **RopCollapseRow** ROP request buffer ([MS-OXCROPS] section 2.2.5.17.1).

CategoryId (8 bytes): An identifier that specifies the category to be collapsed. This field is set to the value of the **PidTagInstID** property (section 2.2.1.1) of the category's header row.

2.2.2.18.2 RopCollapseRow ROP Response Buffer

The following description defines a valid field for the **RopCollapseRow** ROP response buffer ([MS-OXCROPS] section 2.2.5.17.2).

CollapsedRowCount (4 bytes): An unsigned integer that specifies the number of rows that have been collapsed.

2.2.2.19 RopGetCollapseState ROP

The **RopGetCollapseState** ROP ([MS-OXCROPS] section 2.2.5.18) returns the data necessary to rebuild the current expanded/collapsed state of the table. The data returned is in the form of an opaque BLOB that can be passed to a **RopSetCollapseState** ROP request ([MS-OXCROPS] section 2.2.5.19). This ROP is valid only on Table objects.

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

2.2.2.19.1 RopGetCollapseState ROP Request Buffer

The following descriptions define valid fields for the **RopGetCollapseState** ROP request buffer ([MS-OXCROPS] section 2.2.5.18.1).

RowId (8 bytes): An identifier that specifies the row to be preserved as the current cursor, which is returned in the **CollapseState** field of the **RopGetCollapseState** **ROP response**. This field is set to the value of the **PidTagInstID** property (section 2.2.1.1) of the row to be preserved as the current cursor.

RowInstanceNumber (4 bytes): An unsigned integer that is set to the value of the **PidTagInstanceNum** property (section 2.2.1.2) of the row to be preserved as the current cursor, which is returned in the **CollapseState** field of the **RopGetCollapseState** ROP response.

2.2.2.19.2 RopGetCollapseState ROP Response Buffer

The following descriptions define valid fields for the **RopGetCollapseState** ROP response buffer ([MS-OXCROPS] section 2.2.5.18.2).

CollapseStateSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **CollapseState** field.

CollapseState (variable): An array that contains the data necessary for the **RopSetCollapseState** ROP ([MS-OXCROPS] section 2.2.5.19) to rebuild the table's collapsed state, including the current cursor.

2.2.2.20 RopSetCollapseState ROP

The **RopSetCollapseState** ROP ([MS-OXCROPS] section 2.2.5.19) rebuilds a table's collapsed state, which is specified by the data returned from a **RopGetCollapseState** ROP ([MS-OXCROPS] section 2.2.5.18). The **RopSetCollapseState** ROP response contains a **Bookmark** field that references the row that was identified by the **RowId** and **RowInstanceNumber** fields in the **RopGetCollapseState** ROP request. This ROP is valid only on Table objects.

The collapsed state sent to the server need not have been retrieved from the same table to which it is being applied. The table MUST have the same sort and restriction for the ROP to succeed. If the table is not the same table from which the collapse state was retrieved, the value of the **Bookmark** field specified in the **RopSetCollapseState** ROP response will be invalid. The bookmark returned MUST be freed by using the **RopFreeBookmark** ROP ([MS-OXCROPS] section 2.2.5.14).

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

2.2.2.20.1 RopSetCollapseState ROP Request Buffer

The following descriptions define valid fields for the **RopSetCollapseState** ROP request buffer ([MS-OXCROPS] section 2.2.5.19.1).

CollapseStateSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **CollapseState** field.

CollapseState (variable): An array that contains the data that is necessary to rebuild the table's collapsed state. This data is obtained by sending a **RopGetCollapseState** ROP request ([MS-OXCROPS] section 2.2.5.18).

2.2.2.20.2 RopSetCollapseState ROP Response Buffer

The following descriptions define valid fields for the **RopSetCollapseState** ROP response buffer ([MS-OXCROPS] section 2.2.5.19.2).

BookmarkSize (2 bytes): An unsigned integer that specifies the size, in bytes, of the **Bookmark** field.

Bookmark (variable): An array that contains the bookmark data for the row stored as the current cursor, which was passed in the **CollapseState** field of the **RopSetCollapseState** ROP request ([MS-OXCROPS] section 2.2.5.19).

3 Protocol Details

3.1 Client Details

A table is not a static representation of data. Table rows can be modified, moved, created, and deleted while the Table object is in use. Table notifications are used to inform the client of all changes made to the table since it was opened.

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.

Tabular data is retrieved by a client from the server, and then read, navigated, filtered, sorted, and grouped using the Table Object Protocol properties and operations.

When retrieving a table from the server, the set of properties returned for each row can be limited by a column set, and which rows are returned can be limited by a restriction. The order in which rows are returned can be specified by the sort order.

Additionally, tables can categorize rows by column values, and if a table has multivalue properties, it can have multivalue instances, which allows for categorization based on these multivalue instances.

Navigation in a table is aided by using a cursor and bookmarks. The current location in a table is determined by the cursor position, which points to a row of data. A row can also be referred to by a bookmark. Three predefined bookmarks point to the current position, the beginning of a table, and the end of a table, as specified in section 2.2.2.1.1.

A client can have more than one table opened on each data source. Each table is independent and has its own cursor, bookmarks, sort order, restriction, column set, and notification requests.

3.1.2 Timers

None.

3.1.3 Initialization

A client can begin using this protocol with a valid table handle. The method to open the table and acquire a handle is dependent on the table's type, as described in section 1.5.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Preparing the Table

When a higher layer, or the user, needs to modify the column set, the client MUST send a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) to change the column set.

When a higher layer, or the user, needs to modify the sort order, the client MUST send a **RopSortTable** ROP request ([MS-OXCROPS] section 2.2.5.2) to change the sort order. The **RopSortTable** ROP is only supported on contents tables.

In order to categorize a table, the category properties MUST be the first properties in the **SortOrders** array passed to the **RopSortTable** ROP. The number of category properties MUST be set in the **CategoryCount** field passed to **RopSortTable**.

When a higher layer, or the user, needs to filter the rows returned in a **RopQueryRows** ([MS-OXCROPS] section 2.2.5.4), **RopFindRow** ([MS-OXCROPS] section 2.2.5.13), or **RopExpandRow** ([MS-OXCROPS] section 2.2.5.16) ROP response, the client MUST send a **RopRestrict** ROP request ([MS-OXCROPS] section 2.2.5.3) to change the filter applied to the table.

When a higher layer, or the user, needs to clean up the table by removing old sorts, column sets, or restrictions, the client MUST send a **RopResetTable** ROP request ([MS-OXCROPS] section 2.2.5.15). After doing this, the client MUST send a **RopSetColumns** ROP request before sending a **RopQueryRows**, **RopFindRow**, or **RopExpandRow** ROP request.

If a request for a **RopSetColumns**, **RopSortTable**, or **RopRestrict** ROP fails, the client can send a **RopResetTable** ROP request before retrying the failed message.

If a **RopSetColumns** ROP request fails, the client MUST consider the table as invalid and MUST NOT send any other ROPs on it until a successful **RopSetColumns** or **RopResetTable** ROP request is made.

If a **RopSortTable** ROP request fails, the client MUST consider the table to be invalid and MUST NOT send any ROPs on it until a successful **RopSortTable** or **RopResetTable** ROP request is made.

If a **RopRestrict** ROP request fails, the client MUST consider the table to be invalid and MUST NOT send any ROPs on it until a successful **RopRestrict** or **RopResetTable** ROP request is made.

3.1.4.1.1 Asynchronous Table Preparation

The client MAY request that the server perform a **RopSetColumns** ([MS-OXCROPS] section 2.2.5.1), **RopSortTable** ([MS-OXCROPS] section 2.2.5.2), or **RopRestrict** ([MS-OXCROPS] section 2.2.5.3) ROP request asynchronously.<17> In this case, the client MUST NOT request additional asynchronous work to be done until pending asynchronous work is complete or canceled by using a **RopAbort** ROP request ([MS-OXCROPS] section 2.2.5.5). If the client requests additional synchronous work while an existing asynchronous request is pending, the server will respond in one of two ways:

- The server returns ecBusy, as specified in [MS-OXCDATA] section 2.4, and does not perform the requested action.
- The server waits until the first asynchronous action is complete, then completes the synchronous action and sends the ROP response at that time.

When a higher layer, or the user, needs to know the status of pending asynchronous requests, the client MUST get the status using a **RopGetStatus** ROP request ([MS-OXCROPS] section 2.2.5.6). When a higher layer, or the user, needs to abort a pending asynchronous request (to set columns, sort the table, or restrict), the client MUST send a **RopAbort** ROP request. After a successful **RopAbort** ROP request, the client MUST assume the table is in an undefined state and use the **RopResetTable** ROP ([MS-OXCROPS] section 2.2.5.15) before using the table again.

If an asynchronous request to a **RopSetColumns**, **RopSortTable**, or **RopRestrict** ROP fails with the error "ecNotSupported", the client can reattempt the request synchronously.

If a **RopSetColumns** ROP request fails, the client MUST assume that the table has an invalid column set and MUST perform a successful **RopSetColumns** ROP request before proceeding.

If a **RopSortTable** ROP request fails, the client MUST assume that the table has an invalid sort and MUST perform a successful **RopSortTable** ROP request before proceeding.

If a **RopRestrict** ROP request fails, the client MUST assume that the table has an invalid restriction and MUST perform a successful **RopRestrict** ROP request before proceeding.

3.1.4.2 Querying the Table

When a higher layer, or the user, requests tabular data from a table, the client MUST retrieve that information using a **RopQueryRows** ([MS-OXCROPS] section 2.2.5.4) or **RopFindRow** ([MS-OXCROPS] section 2.2.5.13) ROP request.

The client can get the whole table by sending a **RopQueryRows** ROP request repeatedly with the *Advance* option until the **RopQueryRows** ROP response returns zero rows (indicating the end of the table has been reached).

When a higher layer, or the user, needs to know the list of available columns for the table, the client MUST get the column list by sending a **RopQueryColumnsAll** ROP request ([MS-OXCROPS] section 2.2.5.12).

When a higher layer, or the user, needs to collapse rows that are grouped into a category into one header row, the client MUST send a **RopCollapseRow** ROP request ([MS-OXCROPS] section 2.2.5.17).

When a higher layer, or the user, needs to expand rows that are grouped into a collapsed header row, the client MUST send a **RopExpandRow** ROP request ([MS-OXCROPS] section 2.2.5.16). The client can retrieve some or all of the rows expanded.

If the client is going to expand and collapse categories, it MUST include the **PidTagInstID** property (section 2.2.1.1) in the **PropertyTags** field of the **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1), and use the value of that property in **RopExpandRow**, **RopCollapseRow**, and **RopGetCollapseState** ([MS-OXCROPS] section 2.2.5.18) ROP requests.

3.1.4.3 Advancing the Table

When querying the table, the client can advance the table by setting the **QueryRowsFlags** field to 0x00 in the **RopQueryRows** ROP request ([MS-OXCROPS] section 2.2.5.4). Additionally, when higher layers need to move the current cursor in the table, the client MUST use a **RopFindRow** ([MS-OXCROPS] section 2.2.5.13), **RopSeekRow** ([MS-OXCROPS] section 2.2.5.8), **RopSeekRowBookmark** ([MS-OXCROPS] section 2.2.5.9), or **RopSeekRowFractional**<18> ([MS-OXCROPS] section 2.2.5.10) ROP request to advance to the correct row. A **RopFindRow** ROP request can be used to both advance the table and query for the row found at the same time.

The client MUST NOT expect a **RopSeekRowFractional** ROP request to place the cursor in any exact position. It is always an approximation.

When a higher layer, or the user, needs to determine the current location in a table, the client MUST send a **RopQueryPosition** ROP request ([MS-OXCROPS] section 2.2.5.7).

When a higher layer, or the user, needs to save the current location in the table for future use, the client MUST send a **RopCreateBookmark** ROP request ([MS-OXCROPS] section 2.2.5.11) and cache the data from the **Bookmark** field.

When a higher layer, or the user, no longer needs a bookmark that was created using a **RopCreateBookmark** ROP, the client SHOULD send a **RopFreeBookmark** ROP request ([MS-OXCROPS] section 2.2.5.14) with the data from the **Bookmark** field. The client can choose not to send a **RopFreeBookmark** ROP request; however, this can degrade server performance until the table is released via the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

When a higher layer, or the user, needs to move the current table location to a previously created bookmark, the client MUST send a **RopSeekRowBookmark** ROP request.

3.1.4.4 Getting Table State

When a higher layer, or the user, needs to preserve the expanded/collapsed state of the categories, the client MUST send a **RopGetCollapseState** ROP request ([MS-OXCROPS] section 2.2.5.18) and store the BLOB sent in the response for future use.

When a higher layer, or the user, needs to reset the expanded/collapsed state of the categories to a previously cached state, the client MUST send a **RopSetCollapseState** ROP request ([MS-OXCROPS] section 2.2.5.19) with the BLOB returned from a **RopGetCollapseState** ROP response. The bookmark sent in the response to a **RopSetCollapseState** ROP request SHOULD be freed using a **RopFreeBookmark** ROP request ([MS-OXCROPS] section 2.2.5.14). The client can choose not to send a **RopFreeBookmark** ROP request; however, this can degrade server performance until the table is released via the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

3.1.4.5 Registering For Notifications

The Table Object Protocol assumes that tables are dynamic. This means that the state on the server can significantly change while waiting for responses. If the server supports notifications on the table, the client SHOULD register for notifications and respond appropriately so that it has an accurate understanding of server state. For details about notifications, see [MS-OXCNOTIF].

3.1.4.6 Retrieving Conversation Views

To retrieve a list of messages associated with a conversation, a client first calls the **RopGetContentsTable** ROP ([MS-OXCROPS] section 2.2.4.14) with the **TableFlags** field set to ConversationMembers. The client then MUST call the **RopRestrict** ROP ([MS-OXCROPS] section 2.2.5.3) to restrict the table results to a specific conversation. The client then calls the **RopQueryRows** ROP ([MS-OXCROPS] section 2.2.5.4) on the table to obtain the messages included in the specified conversation.

3.1.5 Message Processing Events and Sequencing Rules

After opening the table or sending a **RopResetTable** ROP request ([MS-OXCROPS] section 2.2.5.15), the client MUST send a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) before querying the table for data.

The client SHOULD send a **RopSortTable** ROP request ([MS-OXCROPS] section 2.2.5.2) before querying a contents table for data. If the client does not send a **RopSortTable** ROP request, it MUST consider the sort order of the table as undefined.

The client can send a **RopRestrict** ROP request ([MS-OXCROPS] section 2.2.5.3) before querying the table for data.

When the **TableStatus** field of the **RopSortTable** ROP response has a value that is equal to "TBLSTAT_SORT_ERROR", then the **RopSortTable** ROP failed, and the client MUST consider the table invalid until it receives a successful a **RopSortTable** ROP response. When the **TableStatus** field of the **RopSetColumns** ROP response has a value that is equal to "TBLSTAT_SETCOL_ERROR", then the **RopSetColumns** ROP failed, and the client MUST consider the table invalid until it receives a successful **RopSetColumns** ROP response. When the **TableStatus** field of the **RopRestrict** ROP response has a value that is equal to "TBLSTAT_RESTRICT_ERROR", then the **RopRestrict** ROP failed, and the client MUST consider the table invalid until it receives a successful **RopRestrict** ROP response.

The client SHOULD NOT send a **RopAbort** ROP request ([MS-OXCROPS] section 2.2.5.5) unless the last **TableStatus** field returned in a ROP response indicated that the server is executing an asynchronous task.<19> If the server has no asynchronous work executing when a **RopAbort** ROP is requested, it will return "ecUnableToAbort" in the **ReturnValue** field of the **RopAbort** ROP response.

3.1.5.1 Processing Notifications

The protocol assumes that tables are dynamic. This means that the state on the server can significantly change while waiting for responses. If the server supports notifications on the table, the client SHOULD register for notifications and respond appropriately in order to have an accurate understanding of server state. For details about notifications, see [MS-OXCNOTIF].

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The abstract data model for the server is the same as the abstract data model for the client, as defined in section 3.1.1.

3.2.2 Timers

None

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

The server MUST send notifications to all clients that have requested them based on the appropriate triggers at higher layers. For details about notifications, see [MS-OXCNOTIF].

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Processing Asynchronous Requests

If the client requests that the server perform a **RopSetColumns** ([MS-OXCROPS] section 2.2.5.1), **RopSortTable** ([MS-OXCROPS] section 2.2.5.2), or **RopRestrict** ([MS-OXCROPS] section 2.2.5.3) ROP request asynchronously, the server MAY<20> perform the operation synchronously and return TBLSTAT_COMPLETE in the **TableStatus** field of the ROP response buffer. However, if the server executes the ROP asynchronously, the server SHOULD return TBLSTAT_SORTING, TBLSTAT_SETTING_COLS, or TBLSTAT_RESTRICTING (depending on the ROP performed) in the **TableStatus** field of the ROP response buffer and do the work asynchronously. The server MUST return the same **TableStatus** field value in the **RopGetStatus** ROP response buffer ([MS-OXCROPS]

section 2.2.5.6) unless the work has been completed or a **RopAbort** ROP request ([MS-OXCROPS] section 2.2.5.5) has been sent.

If there is an error setting the columns, sorting the table, or restricting, the next response to a **RopGetStatus** ROP request MUST set the **TableStatus** field to TBLSTAT_SETCOL_ERROR, TBLSTAT_SORT_ERROR, or TBLSTAT_RESTRICT_ERROR, depending on the ROP performed. When the asynchronous work is complete, the server MUST send the **TableRestrictionChanged** or **TableChanged** notifications, depending on the ROP performed. For details about these notifications, see [MS-OXCNOTIF] section 2.2.1.1.1.

If the client requests additional asynchronous work while the server is still performing asynchronous work, the server MUST set the value of the **ReturnValue** field in the ROP response buffer to ecBusy, as specified in [MS-OXCDATA] section 2.4. If the client requests additional synchronous work while the server is still performing asynchronous work, the server can do either of the following:

- Set the value of the **ReturnValue** field in the ROP response buffer to ecBusy and not perform the requested action.
- Wait until the first asynchronous action is complete and then complete the synchronous action and send the ROP response at that time.

3.2.5.2 Processing RopSetColumns

When a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) is received, the server MUST remember the requested columns and apply them to the table when executing other ROPs that act on that table. The columns that are set by the **RopSetColumns** ROP MUST be the ones sent in the responses to subsequent **RopQueryRows** ([MS-OXCROPS] section 2.2.5.4), **RopFindRow** ([MS-OXCROPS] section 2.2.5.13), or **RopExpandRow** ([MS-OXCROPS] section 2.2.5.16) ROPs executed on that table.

If any of the following ROPs are sent before a successful **RopSetColumns** ROP, then the server fails that ROP with a ecNullObject error.<21>

- **RopCollapseRow**
- **RopCreateBookmark**
- **RopExpandRow**
- **RopFindRow**
- **RopFreeBookmark**
- **RopGetCollapseState**
- **RopQueryRows**
- **RopSeekRowBookmark**
- **RopSetCollapseState**

If a **RopSetColumns** ROP request fails, the server SHOULD invalidate the table column set until a successful **RopSetColumns** ROP request is made. The server can restore the previous column set.

If the TBL_ASYNC bit of the **SetColumnsFlags** field is set, the server can execute the ROP as a table-asynchronous ROP,<22> as specified in section 3.2.5.1.

The **RopSetColumns** ROP MUST be supported for all types of tables.

If a column has the **MultivalueInstance** bit set in a **PropertyTag** structure, as specified in [MS-OXCDATA] section 2.9, the server MUST expand the rows that have multiple values for the property

into multivalue instances in subsequent **RopQueryRows**, **RopFindRow**, or **RopExpandRow** ROPs that are executed on the table.

The following specific error codes apply to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecInvalidParam	0x80070057, %x57.00.07.80	A property tag in the column array is of type PT_UNSPECIFIED, PT_ERROR, or an invalid type. These property types are specified in [MS-OXCDATA] section 2.11.1.
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.

3.2.5.3 Processing RopSortTable

When a RopSortTable ROP request ([MS-OXCROPS] section 2.2.5.2) is received, the server MUST apply the sort order to the table, and subsequent requests sent that operate on the table MUST consider the new sort order.

If a sort order is already specified, the new sort order returned with the ROP response MUST replace the old sort order.

When this ROP is sent, the server MUST invalidate all current bookmarks of the table and MUST move the cursor position to the beginning of the table.

If a **RopSortTable** ROP request is not sent (a sort order is not specified), then the table MUST be considered as having the default sort order. Default sort order is undefined.

If the **RopSortTable** ROP fails, the server SHOULD invalidate the table sort order until a successful **RopSortTable** ROP is made. The server can restore the previous sort order.

If the TBL_ASYNC bit of the **SortTableFlags** field is set, the server can execute the ROP as a table-asynchronous ROP,<23> as specified in section 3.2.5.1.

The **RopSortTable** ROP MUST be supported for contents tables.

If a multivalue property has the **MultivalueInstance** bit set in the **SortOrder** structure, the server MUST sort the rows that have multiple values for the property according to the single values used in the multivalue instances in subsequent **RopQueryRows** ([MS-OXCROPS] section 2.2.5.4), **RopFindRow** ([MS-OXCROPS] section 2.2.5.13), or **RopExpandRow** ([MS-OXCROPS] section 2.2.5.16) ROPs that are executed on the table.

The following specific error codes apply to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table.

3.2.5.4 Processing RopRestrict

When a **RopRestrict** ROP request ([MS-OXCROPS] section 2.2.5.3) is received, the server MUST apply the restriction to the table, and subsequent requests that operate on the table MUST consider the new restriction.

If a restriction is applied to a table, the table MUST appear as if it only contains the rows that match the restriction.

When this ROP is sent, the server MUST invalidate all current bookmarks of the table and MUST move the cursor position to the beginning of the table.

If a **RopRestrict** ROP request is not sent (a restriction is not specified), then the table MUST be considered as not having any restrictions.

If a **RopRestrict** ROP fails, the server SHOULD invalidate the table restriction until a successful **RopRestrict** ROP request is made. The server can restore the previous restriction.

If the TBL_ASYNC bit of the **RestrictFlags** field is set, the server MAY<24> execute the ROP as a table-asynchronous ROP, as specified in section 3.2.5.1.

The **RopRestrict** ROP MUST be supported for contents tables, hierarchy tables, and rules tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table, hierarchy table, or rules table.
ecConversationNotFound	0x00000496, %x96.04.00.00	The conversation specified in the restriction does not exist on the server.

3.2.5.5 Processing RopQueryRows

When a **RopQueryRows** ROP request ([MS-OXCROPS] section 2.2.5.4) is sent, the server MUST send the rows from the table starting at the current cursor position.

The number of rows sent in the ROP response MUST be less than or equal to the number of rows specified in the **RowCount** field. The number of rows sent in the response MUST be as many rows as can fit in the ROP response buffer. Whole rows MUST always be sent (partial rows MUST NOT be sent). If there are rows to send in the database, at least one row MUST be returned or the ROP MUST fail with "ecBufferTooSmall", as specified in [MS-OXCDATA] section 2.4. This ROP MUST only send zero rows when there are no more rows in the table.

The **Origin** field in a successful ROP response will have one the predefined values specified in section 2.2.2.1.1. If there are no more rows to return and the **ForwardRead** field in the ROP request is set to "TRUE", then the **Origin** field is set to "BOOKMARK_END". If there are no more rows to return and the **ForwardRead** field in the ROP request is set to "FALSE", then the **Origin** field SHOULD<25> be set to "BOOKMARK_BEGINNING". Otherwise, it will be set to "BOOKMARK_CURRENT".

If rows are returned by the **RopQueryRows** ROP and the **ForwardRead** field is set to "TRUE" (0x01), the **RopQueryRows** ROP returns the rows beginning at the position pointed to by the **Origin** field, reading forward. If it is set to "FALSE" (0x00), the server returns the rows starting at the number of rows equal to the value of the **RowCount** field before the position pointed to by the **Origin** field, such

that the rows returned are those between the value of the **Origin** field minus the value of the **RowCount** field and the position pointed to by the **Origin** field.

If the **NoAdvance** flag (0x01) is set in the **QueryRowsFlags** field, the server MUST NOT change the position of the cursor.

The **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1) MUST be sent on the table before sending a **RopQueryRows** ROP request. The columns sent in the response for each row MUST be the ones specified on the **RopSetColumns** ROP request. The server MUST complete all asynchronous table ROPs before executing this ROP or fail the ROP with "ecBusy", as specified in section 3.2.5.1.

The **RopQueryRows** ROP MUST be supported for all types of tables.

The following specific error codes apply to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNullObject	0x000004B9, %xB9.04.00.00	**RopSetColumns** has not been sent on this table.
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.
ecBufferTooSmall	0x0000047D, %x7D.04.00.00	The space allocated in the return buffer is insufficient to fit at least one row of data.

3.2.5.6 Processing RopAbort

The **RopAbort** ROP ([MS-OXCROPS] section 2.2.5.5) MUST abort the current asynchronous table ROP that is executing on the table or send an error if there is nothing to abort or if it fails to abort.

If the server receives a **RopAbort** ROP request while asynchronous work is being done, it MUST abort that work. The table state after a **RopAbort** ROP request is received is undefined until the server receives a **RopResetTable** ROP request ([MS-OXCROPS] section 2.2.5.15). This is true whether or not the **RopAbort** ROP succeeds. For more details about asynchronous processing, see section 3.2.5.1.

The **RopAbort** ROP MUST be supported for contents tables and hierarchy tables.

The following specific error codes apply to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecUnableToAbort	0x80040114, %x14.01.04.80	There were no asynchronous operations to abort, or the server was unable to abort the operations.
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table or a hierarchy table.

3.2.5.7 Processing RopGetStatus

The **RopGetStatus** ROP ([MS-OXCROPS] section 2.2.5.6) MUST send the status of the current asynchronous execution being performed on the table in the response,<26> as specified in section 3.2.5.1.

The **RopGetStatus** ROP MUST be supported for all types of tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.

3.2.5.8 Processing RopQueryPosition

The **RopQueryPosition** ROP ([MS-OXCROPS] section 2.2.5.7) MUST send the current position of the cursor and the total number of rows in the table in the response.

The server MUST complete all asynchronous table ROPs before executing this ROP or fail the ROP with "ecBusy", as specified in section 3.2.5.1.

The **RopQueryPosition** ROP MUST be supported for all types of tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.

3.2.5.9 Processing RopSeekRow

The **RopSeekRow** ROP ([MS-OXCROPS] section 2.2.5.8) MUST move the cursor position according to its request fields. If moving the cursor the number of rows equal to the value of the **RowCount** field would put it past the end (or beginning, if seeking backwards) of the table, and the **WantRowMovedCount** field is set to "TRUE" (0x01) in the request, the server MUST set the **HasSoughtLess** field to "TRUE" (0x01) and set the **RowsSought** field to the actual number of rows moved to reach the end of the table (or, in the case of seeking backwards, to the beginning of the table).

The server MUST complete all asynchronous table ROPs before executing this ROP or fail the ROP with "ecBusy",<27> as specified in section 3.2.5.1.

The **RopSeekRow** ROP MUST be supported for all types of tables.

The following specific error code applies to this ROP. If the server encounters this error while processing this ROP, it MUST NOT process any remaining ROPs in the ROP input buffer, as specified in [MS-OXCROPS] section 2.2.1. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code n7ame	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.

3.2.5.10 Processing RopSeekRowBookmark

The **RopSeekRowBookmark** ROP ([MS-OXCROPS] section 2.2.5.9) MUST move the cursor position according to its request fields. It acts in the same way as the **RopSeekRow** ROP ([MS-OXCROPS] section 2.2.5.8), except that it moves the cursor using a custom bookmark, as specified in section 2.2.2.1.2, as a reference.

If the bookmark has become invalid because of a **RopSortTable** ([MS-OXCROPS] section 2.2.5.2), **RopRestrict** ([MS-OXCROPS] section 2.2.5.3), or **RopResetTable** ([MS-OXCROPS] section 2.2.5.15) ROP request, the server SHOULD<28> set the **ReturnValue** field to "ecInvalidBookmark". If the bookmark points to a row that is no longer visible (for example, it has been deleted, or its properties have changed so that it no longer matches the restriction, or its header row has been collapsed), the server MUST set the **RowNoLongerVisible** field to "TRUE" (0x01) and move the cursor to the next row in the table.

The server MUST complete all asynchronous table ROPs before executing this ROP or fail the ROP with "ecBusy", as specified in section 3.2.5.1.

The **RopSeekRowBookmark** ROP MUST be supported for contents tables and hierarchy tables.

The following specific error codes apply to this ROP. If the server encounters one of these errors while processing this ROP, it MUST NOT process any remaining ROPs in the ROP input buffer, as specified in [MS-OXCROPS] section 2.2.1. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecInvalidBookmark	0x80040405, %x05.04.04.80	The bookmark sent in the request is no longer valid.
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table or hierarchy table.

3.2.5.11 Processing RopSeekRowFractional

The **RopSeekRowFractional** ROP ([MS-OXCROPS] section 2.2.5.10) MUST move the cursor position in the same way the **RopSeekRow** ROP ([MS-OXCROPS] section 2.2.5.8) does, except that the desired position is indicated as a fraction of the total table size.<29>

If the **Numerator** field is set to 0, the cursor MUST move to the beginning of the table.

If the **Numerator** field value is greater than or equal to the **Denominator** field, the cursor MUST move to the end of the table.

The cursor MUST be moved to the place in the table that would be closest to the fraction provided. The exact location depends on the implementation of the server.

The server MUST complete all asynchronous table ROPs prior to executing this ROP or fail the ROP with "ecBusy",<30> as specified in section 3.2.5.1.

The **RopSeekRowFractional** ROP MUST be supported for all types of tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.

3.2.5.12 Processing RopCreateBookmark

When the server receives a **RopCreateBookmark** ROP request ([MS-OXCROPS] section 2.2.5.11), it MUST create a custom bookmark, as specified in section 2.2.2.1.2, that uniquely identifies a row in the table and can be subsequently used in the **RopSeekRowBookmark** ROP ([MS-OXCROPS] section 2.2.5.9).

The server can allocate resources on the server to keep track of the bookmark created by **RopCreateBookmark** ROP.

If the client does not send a **RopFreeBookmark** ROP request ([MS-OXCROPS] section 2.2.5.14), the server MUST release all bookmarks related to a table when that table is released as a result of sending a **ROPRelease** ROP request ([MS-OXCROPS] section 2.2.15.3) on the table or when a **RopResetTable** ([MS-OXCROPS] section 2.2.5.15), **RopSortTable** ([MS-OXCROPS] section 2.2.5.2), or **RopRestrict** ([MS-OXCROPS] section 2.2.5.3) ROP request is sent.

Before sending a **RopCreateBookmark** ROP request, a client SHOULD send a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1).<31>

The server MUST complete all asynchronous table ROPs prior to executing this ROP or fail the ROP with "ecBusy",<32> as specified in section 3.2.5.1.

The **RopCreateBookmark** ROP MUST be supported for contents tables and hierarchy tables.

The following specific error code applies to this ROP. If the server encounters this error while processing this ROP, it MUST NOT process any remaining ROPs in the ROP input buffer, as specified in [MS-OXCROPS] section 2.2.1. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table or hierarchy table.

3.2.5.13 Processing RopQueryColumnsAll

The **RopQueryColumnsAll** ROP ([MS-OXCROPS] section 2.2.5.12) MUST send all properties that can be queried for in the table in the response.

The **RopQueryColumnsAll** ROP MUST be supported for all types of tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Meaning
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.

3.2.5.14 Processing RopFindRow

The **RopFindRow** ROP ([MS-OXCROPS] section 2.2.5.13) sets the cursor position to the first row that matches the search criteria specified in the ROP (starting the search from the current cursor position) and returns the found row when there is enough space in the output buffer. Rows that do not match the search criteria MUST remain unchanged in the table, but the cursor position is updated to match the found row. If no rows match the search criteria, the cursor position for the **RopFindRow** ROP is undefined.

If the client requested that the find be performed from a custom bookmark, as specified in section 2.2.2.1.2, but the bookmark has become invalid because of a **RopSortTable** ([MS-OXCROPS] section 2.2.5.2), **RopRestrict** ([MS-OXCROPS] section 2.2.5.3), or **RopResetTable** ([MS-OXCROPS] section 2.2.5.15) ROP request, then the server SHOULD<33> set the **ReturnValue** field to "ecInvalidBookmark". If the bookmark points to a row that is no longer visible (for example, the row has been deleted, or its properties have changed so that it no longer matches the restriction, or its header row has been collapsed), the server MUST set the **RowNoLongerVisible** field to "TRUE" (0x01), and perform the find from the next row in the table.

A **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) MUST be sent on the table before sending a **RopFindRow** ROP request. The columns sent for the row found MUST be the columns that are specified on the **RopSetColumns** ROP.

The server MUST complete all asynchronous table ROPs before executing this ROP or fail the ROP with "ecBusy",<34> as specified in section 3.2.5.1.

The **RopFindRow** ROP MUST be supported on contents tables, hierarchy tables, and rules tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table, hierarchy table, or rules table.

3.2.5.15 Processing RopFreeBookmark

The **RopFreeBookmark** ROP ([MS-OXCROPS] section 2.2.5.14) MUST release any resources on the server used to keep track of the bookmark (created using a **RopCreateBookmark** ROP ([MS-OXCROPS] section 2.2.5.11)).

If the client does not send a **RopFreeBookmark** ROP request, the server MUST release all bookmarks related to a table if that table is released as a result of sending a **RopRelease** ROP request ([MS-OXCROPS] section 2.2.15.3) on the table.

The **RopFreeBookmark** ROP MUST be supported by hierarchy tables.

The **RopFreeBookmark** ROP SHOULD be supported by contents tables.<35>

The following specific error codes apply to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102 %x02.01.04.80	The object on which this ROP was sent is not a hierarchy table.
ecNullObject	0x000004B9 %xB9.04.00.00	Attempted to use the bookmark after it was released by the **ROPRelease** ROP ([MS-OXCROPS] section 2.2.15.3).
ecInvalidBookmark	0x80040405 %x05.04.04.80	Attempted to use the bookmark after it was released by the **RopFreeBookmark** ROP (section 2.2.2.15).

3.2.5.16 Processing RopResetTable

The **RopResetTable** ROP ([MS-OXCROPS] section 2.2.5.15) MUST remove the column set previously specified by a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) (if any). The columns on the table MUST be treated as if a **RopSetColumns** ROP request had never been sent on the table.

The **RopResetTable** ROP MUST remove the restriction previously applied to the table using **RopRestrict** ROP ([MS-OXCROPS] section 2.2.5.3) (if any). The table MUST afterwards appear as if the **RopRestrict** ROP had never been sent on it; that is, as if it had no restriction (all rows MUST be present).

The **RopResetTable** ROP MUST remove the sort order previously applied to the table using the **RopSortTable** ROP ([MS-OXCROPS] section 2.2.5.2) (if any). The table MUST afterwards appear as if a **RopSortTable** ROP had never been sent on it. (The default sort order is undefined.)

The **RopResetTable** ROP MUST clear any errors that currently invalidate the table (if any), such as a failed send to a **RopSortTable** or a **RopRestrict** ROP. Note that even though the errors for the table are cleared, it is not ready to accept a **RopQueryRows** ROP ([MS-OXCROPS] section 2.2.5.4), because its column set has not been specified. (The column set can be specified by sending a **RopSetColumns** ROP request).

The **RopResetTable** ROP MUST move the cursor to the beginning of the table.

After a **RopResetTable** ROP executes, all previously existing bookmarks on the table are invalid.

The server MUST complete all asynchronous table ROPs prior to executing this ROP or fail the ROP with "ecBusy",<36> as specified in section 3.2.5.1.

The **RopResetTable** ROP MUST be supported on contents tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table.

3.2.5.17 Processing RopExpandRow

The **RopExpandRow** ROP ([MS-OXCROPS] section 2.2.5.16) sets a category row to expanded state.

The **RopExpandRow** ROP MUST be supported for contents tables.

The following specific error codes apply to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotFound	0x8004010F, %x0F.01.04.80	The row specified by the **CategoryId** field was not found.
ecNotCollapsed	0x000004F8, %xF8.04.00.00	The row specified by the **CategoryId** field was not collapsed.
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table.

3.2.5.18 Processing RopCollapseRow

The **RopCollapseRow** ROP ([MS-OXCROPS] section 2.2.5.17) MUST set a category row to collapsed state.

The **RopCollapseRow** ROP MUST be supported for contents tables.

The following specific error codes apply to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotFound	0x8004010F, %x0F.01.04.80	The row specified by the **CategoryId** field was not found.
ecNotExpanded	0x000004F7, %xF7.04.00.00	The row specified by the **CategoryId** field was not expanded.
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table.

3.2.5.19 Processing RopGetCollapseState

The **RopGetCollapseState** ROP ([MS-OXCROPS] section 2.2.5.18) MUST send the collapsed state of the whole table in the **CollapseState** field of the ROP response. The collapsed state indicates what categories are expanded. It MUST also include a bookmark to the row indicated by the **RowId** and **RowInstanceNumber** fields.

The **RopGetCollapseState** ROP MUST be supported for contents tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table.

3.2.5.20 Processing RopSetCollapseState

The **RopSetCollapseState** ROP ([MS-OXCROPS] section 2.2.5.19) MUST modify the collapsed state of the table to match the collapsed state being sent. The collapsed state indicates what categories are expanded. It MUST also move the cursor position to the row specified by the bookmark.

The **RopSetCollapseState** ROP MUST be supported for contents tables.

The following specific error code applies to this ROP. For more details about ROP errors returned, see [MS-OXCDATA] section 2.4.

Error code name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not a contents table.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

The following examples illustrate the byte order of ROPs in a buffer being prepared for transmission. Please note that the examples listed here only show the relevant portions of the specified ROPs; this is not the final byte sequence that gets transmitted over the wire. Also note that the data for a multibyte field appear in **little-endian** format, with the bytes in the field presented from least significant to most significant. Generally speaking, these ROP requests are compressed and packed with other ROP requests as described in [MS-OXCRPC]. These examples assume the client has already successfully logged on to the server and opened the table. Unless otherwise noted, these examples are additive; the second example is performed after the first example, and so on. For more information, see [MS-OXCROPS].

Examples in this section use the following format for byte sequences:

0080: 45 4d 53 **4d** 44 42 2e 44-4c 4c 00 00 00 00 00 00

The bold value at the far left is the offset of the following bytes into the buffer, expressed in hexadecimal notation. Following the offset is a series of up to 16 bytes, with each two character sequence describing the value of 1 byte in hexadecimal notation. Here, the bold byte "4d" (01001101) is located 0x83 bytes (131 bytes) from the beginning of the buffer. The dash between the eighth byte ("44") and the ninth byte ("4c") has no semantic value and serves only to distinguish the 8-byte boundary for readability purposes.

Such a byte sequence is then followed by one or more lines interpreting it. In larger examples, the byte sequence is shown once in its entirety and then repeated in smaller chunks, with each smaller chunk interpreted separately.

The following example shows how a property tag and its property value are represented in a buffer and interpreted directly from it (according to the **TaggedPropertyValue** structure format described in [MS-OXCDATA] section 2.11.4). The property tag appears in the buffer in little-endian format.

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

PropertyTag: 0x66760003 (**PidTagRuleSequence** ([MS-OXPROPS] section 2.953))

PropertyValue: 10

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

4.1 Obtaining a Message List

The following example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopGetContentsTable** operation ([MS-OXCROPS] section 2.2.4.14), as described in [MS-OXCFOLD] section 2.2.1.14.

4.1.1 Client Request Buffer

A complete ROP request buffer is a 5-byte sequence formatted as follows:

```
0000: 05 00 00 01 00
```

The first 4 bytes are the **RopId**, **LogonID**, **InputHandleIndex**, and **OutputHandleIndex** fields, as described in [MS-OXCROPS] section 2.2.4.14.1.

```
0000: 05 00 00 01
```

RopId: 0x05 (**RopGetContentsTable** ([MS-OXCROPS] section 2.2.4.14))

LogonID: 0x00

InputHandleIndex: 0x00. The object for which to obtain the contents table (such as a **Folder object**).

OutputHandleIndex: 0x01. The location to store the table.

The last byte is the **TableFlags** field, which holds the table operation flags (described in [MS-OXCFOLD] section 2.2.1.14).

```
0004: 00
```

TableFlags: 0x00 (Standard)

4.1.2 Server Response to Client Request

0000: 05 01 00 00 00 00 04 00-00 00

The first 6 bytes of the ROP response buffer are the **RopId**, **OutputHandleIndex**, and **ReturnValue** fields, as described in [MS-OXCROPS] section 2.2.4.14.2.

```
0000: 05 01 00 00 00 00
```

RopId: 0x05 (**RopGetContentsTable** ([MS-OXCROPS] section 2.2.4.14))

OutputHandleIndex: 0x01

ReturnValue: 0x00000000 (Success)

The next 4 bytes are the **RowCount** field, as described in [MS-OXCFOLD] section 2.2.1.14.2, which gives the number of rows in the contents table.

```
0006: 04 00-00 00
```

RowCount: 0x00000004 (four rows in the table)

4.2 Setting the Columns on a Table

The following example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopSetColumns** operation ([MS-OXCROPS] section 2.2.5.1), as specified in section 2.2.2.2.

4.2.1 Client Request Buffer

A complete ROP request buffer is a variable length sequence, with 6 required bytes and 4 bytes for each property tag to be included in the columns set. An example of the ROP request buffer is as follows:

```
0000: 12 00 01 00 06 00 14 00-48 67 14 00 4a 67 14 00
0010: 4d 67 03 00 4e 67 1f 00-37 00 40 00 06 0e
```

The first 3 bytes of the ROP request buffer are the **RopId**, **LogonID**, and **InputHandleIndex** fields of the **RopSetColumns** ROP, as described in [MS-OXCROPS] section 2.2.5.1.1.

```
0000: 12 00 01
```

RopId: 0x12 (**RopSetColumns**)

LogonID: 0x00

InputHandleIndex: 0x01

The next 3 bytes are the **SetColumnsFlags** and **PropertyTagCount** fields of the **RopSetColumns** ROP, described in section 2.2.2.2.1. For more information on property buffer format, see [MS-OXCDATA].

```
0003: 00 06 00
```

SetColumnsFlags: 0x00. Perform this operation synchronously.

PropertyValueCount: 0x0006. Six 4-byte property tags follow.

The remaining bytes are the **PropertyTags** field, which holds an array of 4-byte property tags.

```
0006: 14 00 48 67 14 00 4a 67-14 00 4d 67 03 00 4e 67
0016: 1f 00 37 00 40 00 06 0e
```

PropertyTag: 0x67480014 (**PidTagFolderId** ([MS-OXCFOLD] section 2.2.2.2.1.6))

PropertyTag: 0x674a0014 (**PidTagMid** ([MS-OXCFXICS] section 2.2.1.2.1))

PropertyTag: 0x674d0014 (**PidTagInstID** (section 2.2.1.1))

PropertyTag: 0x674e0003 (**PidTagInstanceNum** (section 2.2.1.2))

PropertyTag: 0x0037001f (**PidTagSubject** ([MS-OXPROPS] section 2.1023))

PropertyTag: 0x0e060040 (**PidTagMessageDeliveryTime** ([MS-OXOMSG] section 2.2.3.9))

4.2.2 Server Response to Client Request

0000: 12 01 00 00 00 00 00

The first 6 bytes of the ROP response buffer are the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

RopId: 0x12 (**RopSetColumns** ([MS-OXCROPS] section 2.2.5.1))

InputHandleIndex: 0x01

ReturnValue: 0x00000000 (Success)

The final byte in the ROP response buffer is the **TableStatus** field, described in section 2.2.2.1.3.

```
0006: 00
```

TableStatus: 0x00. This value is "TBLSTAT_COMPLETE", indicating that the operation has been completed.

4.3 Sorting a Table by Time Delivered

The following example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopSortTable** operation ([MS-OXCROPS] section 2.2.5.2) as described in section 2.2.2.3.

4.3.1 Client Request Buffer

A complete ROP request buffer is a variable length sequence, with 10 required bytes and 5 bytes for each sorting flag used. An example of the ROP request buffer is as follows:

```
0000: 13 00 01 00 01 00 00 00-00 00 40 00 06 0e 01
```

The first 3 bytes of the ROP request buffer are the **RopId**, **LogonID**, and **InputHandleIndex** fields of the **RopSetColumns** ROP, as described in [MS-OXCROPS] section 2.2.5.1.1.

```
0000: 13 00 01
```

RopId: 0x13 (**RopSortTable** ([MS-OXCROPS] section 2.2.5.2))

LogonID: 0x00

InputHandleIndex: 0x01. Index in **handle array** for the table to be sorted.

The next 7 bytes are the **SortTableFlags**, **SortOrderCount**, **CategoryCount**, and **ExpandedCount** fields described in section 2.2.2.3.1.

```
0003: 00 01 00 00 00 00 00
```

SortTableFlags: 0x00. Perform the operation synchronously.

SortOrderCount: 0x0001. Number of **SortOrder** structures to follow

CategoryCount: 0x0000

ExpandedCount: 0x0000

The remaining bytes are the **SortOrders** field, which contain properties to sort by (where there must be exactly the number of properties specified by the **SortOrderCount** field) and a sorting method, as described in section 2.2.2.3.1.

```
000a: 40 00 06 0e 01
```

PropertyTag: 0x0e060040 (**PidTagMessageDeliveryTime** ([MS-OXOMSG] section 2.2.3.9))

Order: 0x01 (Flag: TABLE_SORT_DESCEND)

4.3.2 Server Response to Client Request

0000: 13 01 00 00 00 00

The first 6 bytes of the ROP response buffer are the **RopId**, **InputHandleIndex**, and **ReturnValue** fields.

```
0000: 13 01 00 00 00 00
```

RopId: 0x13 (**RopSortTable** ([MS-OXCROPS] section 2.2.5.2))

InputHandleIndex: 0x01

ReturnValue: 0x00000000 (Success)

The final byte in the ROP response buffer is the **TableStatus** field described in section 2.2.2.3.2.

```
0006: 00
```

TableStatus: 0x00. This value is "TBLSTAT_COMPLETE", indicating that the sorting operation has been completed.

4.4 Querying Rows

The following example describes the contents of the ROP request buffer and ROP response buffer for a successful **RopQueryRows** operation ([MS-OXCROPS] section 2.2.5.4), as described in section 2.2.2.5.

4.4.1 Client Request Buffer

A complete ROP request buffer for the **RopQueryRows** ROP ([MS-OXCROPS] section 2.2.5.4) is a 7-byte sequence formatted as follows:

```
0000: 15 00 01 00 01 32 00
```

The first 3 bytes are the **RopId**, **LogonId**, and **InputHandleIndex** fields as described in [MS-OXCROPS] section 2.2.5.4.1.

```
0000: 15 00 01
```

RopId: 0x15 (**RopQueryRows**)

LogonID: 0x00

InputHandleIndex: 0x01. The handle of the table to query.

The final 4 bytes of the ROP request buffer are the **QueryRowsFlags**, **ForwardRead**, and **RowCount** fields described in section 2.2.2.5.1.

```
0003: 00 01 32 00
```

QueryRowsFlags: 0x00. Advance the table cursor.

ForwardRead: 0x01. Read the table forward.

RowCount: 0x0032. Return a maximum of 50 rows.

4.4.2 Server Response to Client Request

0000: 15 01 00 00 00 00 02 04-00 01 00 01 00 00 00 00

```
0010: 00 14 88 00 01 00 00 00-0b 3f 87 47 00 01 00 00
0020: 00 0b 3f 87 47 00 00 00-00 00 00 01 00 00 00 0a
0030: 0f 01 04 80 00 49 00 50-00 4d 00 2e 00 4e 00 6f
0040: 00 74 00 65 00 00 00 00-ff ff ff ff 00 00 00 00
0050: 00 0a 0f 01 04 80 00 00-00 00 00 00 23 00 00 00
0060: 0a 0f 01 04 80 0a 0f 01-04 80 0a 0f 01 04 80 0a
0070: 0f 01 04 80 0a 0f 01 04-80 0a 0f 01 04 80 00 ea
0080: 04 00 00 00 1d 05 03 ea-55 73 c8 01 00 6d 00 79
0090: 00 53 00 75 00 62 00 6a-00 65 00 63 00 74 00 00
00a0: 00 00 52 00 45 00 3a 00-20 00 6d 00 79 00 53 00
00b0: 75 00 62 00 6a 00 65 00-63 00 74 00 00 00 00 41
00c0: 00 75 00 74 00 6f 00 55-00 73 00 65 00 72 00 32
00d0: 00 30 00 30 00 30 00 31-00 00 00 0a 0f 01 04 80
00e0: 0a 0f 01 04 80 00 00 00-00 00 01 00 00 00 00 10
00f0: 00 f6 e9 ad 14 41 50 e6-4d 9f 42 64 6e d0 98 c2
```

The first 6 bytes of the ROP response buffer are the **RopId**, **InputHandleIndex**, and **ReturnValue** fields, as described in [MS-OXCROPS] section 2.2.5.4.2.

```
0000: 15 01 00 00 00 00
```

RopId: 0x15 (**RopQueryRows** ([MS-OXCROPS] section 2.2.5.4))

InputHandleIndex: 0x01

ReturnValue: 0x00000000 (Success)

The next 3 bytes are the **Origin** and **RowCount** fields described in section 2.2.2.5.2.

```
0006: 02 04 00
```

Origin: 0x02. Corresponds to the predefined **Bookmark** field value "BOOKMARK_END", as described in section 2.2.2.1.1.

RowCount: 0x0004. Four **FlaggedPropertyValue** structures follow in the response.

The remaining bytes in the ROP response buffer are for the **RowData** array, which consists of a series of flags for the row coupled with a **FlaggedPropertyValue** field, as described in [MS-OXCDATA] section 2.11.5.

```
0009: 01 00 01 00 00 00 00 00-14 88 00 01 00 00 00 0b
0019: 3f 87 47 ...
```

Has Flag: 0x01 (for the row)

Flag: 0x00

FlaggedPropertyValue: 0x8814000000000001. From the **RopSetColumns** operation, this property is **PidTagFolderId** ([MS-OXCFOLD] section 2.2.2.2.1.6) (0x67480014) because the order must be maintained.

Flag: 0x00

FlaggedPropertyValue: 0x47873f0b00000001. From the **RopSetColumns** operation, this property is **PidTagMid** ([MS-OXCFXICS] section 2.2.1.2.1) (0x674a0014) because the order must be maintained.

This format continues for the remainder of the column properties and then for the remainder of the rows.

4.5 Working with Categories

The following sections give examples of sorting, expanding a row, and querying messages that have been categorized. These examples are separate and do not follow from the ones above.

4.5.1 Sorting a Table by Category

The following example describes the contents of the ROP request buffer and ROP response buffers for a successful **RopSortTable** operation ([MS-OXCROPS] section 2.2.5.2) for a category (ascending) and a time (descending) sort as described in section 2.2.2.3.

4.5.1.1 Client Request Buffer

A complete ROP request buffer is a variable length sequence, with 10 required bytes and 5 bytes for each sorting flag used. An example of the ROP request buffer is as follows:

```
0000: 13 00 00 00 02 00 01 00-01 00 1f 30 08 80 00 40
0010: 00 06 0e 01
```

The first 3 bytes of the ROP request buffer are the **RopId**, **LogonID**, and **InputHandleIndex** fields of the **RopSortTable** ROP, as described in [MS-OXCROPS] section 2.2.5.2.1.

```
0000: 13 00 00
```

RopId: 0x13 (**RopSortTable**)

LogonID: 0x00

InputHandleIndex: 0x00. Index in handle array for the table to be sorted.

The next 7 bytes are the **SortTableFlags**, **SortOrderCount**, **CategoryCount**, and **ExpandedCount** fields defined in section 2.2.2.3.1.

```
0003: 00 02 00 01 00 01 00
```

SortTableFlags: 0x00. Perform the operation synchronously.

SortOrderCount: 0x0002. Number of sort order structures to follow.

CategoryCount: 0x0001. There is one category column.

ExpandedCount: 0x0001. All categories are expanded.

The remaining bytes are the **SortOrders** field, which contain properties to sort by (where there has to be exactly the number of properties given in the **SortOrderCount** field) and a sorting method (defined in section 2.2.2.3.1).

```
000a: 1f 30 08 80 00 40 00 06-0e 01
```

PropertyTag: 0x8008301F (**PidTagAddressBookIsMemberOfDistributionList** ([MS-OXOABK] section 4.5.3.2))

Order: 0x00 (Flag: TABLE_SORT_ASCEND)

PropertyTag: 0x0E060040 (**PidTagMessageDeliveryTime** ([MS-OXOMSG] section 2.2.3.9)

Order: 0x01 (Flag: TABLE_SORT_DESCEND)

4.5.1.2 Server Response to Client Request

0000: 13 00 00 00 00 00 00

The first 6 bytes of the ROP response buffer are the **RopId**, **InputHandleIndex**, and **ReturnValue** fields, as described in [MS-OXCROPS] section 2.2.5.2.2.

```
0000: 13 00 00 00 00 00
```

RopId: 0x13 (**RopSortTable** ([MS-OXCROPS] section 2.2.5.2))

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

The final byte in the ROP response buffer is the **TableStatus** field described in section 2.2.2.3.2.

```
0006: 00
```

TableStatus: 0x00. This value is "TBLSTAT_COMPLETE", indicating that the sorting operation has been completed.

4.5.2 Expanding a Category Row

The following example describes the contents of the ROP request buffer and ROP response buffers for a successful **RopExpandRow** ([MS-OXCROPS] section 2.2.5.16) operation as described in section 2.2.2.17.

4.5.2.1 Client Request Buffer

A complete ROP request buffer is a 13-byte sequence, formatted as follows:

```
0000: 59 00 01 00 00 01 00 00-00 00 f1 88 bd
```

The first 3 bytes of the ROP request buffer are the **RopId**, **LogonID**, and **InputHandleIndex** fields of the **RopExpandRow** ROP, as described in [MS-OXCROPS] section 2.2.5.16.1.

```
0000: 59 00 00
```

RopId: 0x59 (**RopExpandRow** ([MS-OXCROPS] section 2.2.5.16))

LogonID: 0x00

InputHandleIndex: 0x01

The remaining 10 bytes are the **MaxRowCount** and **CategoryId** fields described in section 2.2.2.17.1.

```
0003: 00 00 01 00 00 00 00 f1-88 bd
```

MaxRowCount: 0x0000. Rows will be expanded but not returned in the response.

CategoryId: 0xbd88f10000000001. The **PidTagInstID** (section 2.2.1.1) of the category row to expand.

4.5.2.2 Server Response to Client Request

0000: 59 01 00 00 00 00 03 00-00 00 00 00

The first 6 bytes of the ROP response buffer are the **RopId**, **InputHandleIndex**, and **ReturnValue** fields, as described in [MS-OXCROPS] section 2.2.5.16.2.

```
0000: 59 00 00 00 00 00
```

RopId: 0x59 (**RopExpandRow** ([MS-OXCROPS] section 2.2.5.16))

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

The remaining bytes are the **ExpandedRowCount**, **RowCount**, and **RowData** fields described in section 2.2.2.17.2.

```
0006: 03 00 00 00 00 00
```

ExpandedRowCount: 0x00000003. There are a total of three rows in the expanded category.

RowCount: 0x0000. No row data follows.

RowData: [EMPTY]

4.5.3 Querying Rows with Category View

The following example describes the contents of the ROP request buffer and ROP response buffers for a successful **RopQueryRows** operation ([MS-OXCROPS] section 2.2.5.4), as described in section 2.2.2.17, when the messages are grouped by category.

4.5.3.1 Client Request Buffer

A complete ROP request buffer for the **RopQueryRows** ROP ([MS-OXCROPS] section 2.2.5.4) is a 7-byte sequence formatted as follows:

```
0000: 15 00 00 00 01 32 00
```

The first 3 bytes are the **RopId**, **LogonID**, and **InputHandleIndex** fields, as described in [MS-OXCROPS] section 2.2.5.4.1.

```
0000: 15 00 00
```

RopId: 0x15 (**RopQueryRows**)

LogonID: 0x00

InputHandleIndex: 0x00. The handle of the table to query.

The final 4 bytes of the ROP request buffer are the **QueryRowsFlags**, **ForwardRead**, and **RowCount** fields described in section 2.2.2.5.1.

```
0003: 00 01 32 00
```

QueryRowsFlags: 0x00. Advance the table cursor.

ForwardRead: 0x01. Read the table forward.

RowCount: 0x0032. Return a maximum of 50 rows.

4.5.3.2 Server Response to Client Request

0000: 15 00 00 00 00 00 02 09-00 01 00 01 00 00 00 00

```
0010: f1 1f 32 0a 0f 01 04 80-00 01 00 00 00 00 f1 88
0020: bd 00 00 00 00 00 00 03-00 00 00 00 00 00 00 00
0030: 0a 0f 01 04 80 0a 0f 01-04 80 0a 0f 01 04 80 0a ...
```

The first 6 bytes of the ROP response buffer are the **RopId**, **InputHandleIndex**, and **ReturnValue** fields, as described in [MS-OXCROPS] section 2.2.5.4.2.

```
0000: 15 00 00 00 00 00
```

RopId: 0x15 (**RopQueryRows** ([MS-OXCROPS] section 2.2.5.4))

InputHandleIndex: 0x00

ReturnValue: 0x00000000 (Success)

The next 3 bytes are the **Origin** and **RowCount** fields described in section 2.2.2.5.2.

```
0006: 02 09 00
```

Origin: 0x02. Corresponds to the predefined **Bookmark** field value "BOOKMARK_END", as described in section 2.2.2.1.1.

RowCount: 0x0009. Nine **FlaggedPropertyValue** structures follow in the response.

The remaining bytes in the ROP response buffer are for the **RowData** array, which consists of a series of flags for the row coupled with a **FlaggedPropertyValue** field, as described in [MS-OXCDATA] section 2.11.5. The **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) for this sequence of ROPs has not been shown.

```
0009: 01 00 01 00 00 00 00 f1-1f 32 0a 0f 01 04 80 00
0019: 01 00 00 00 00 f1 88 bd-00 00 00 00 00 00 03 00
0029: 00 00 00 00 00 00 00 0a-0f 01 04 80 0a 0f 01 04
0039: 80 0a 0f 01 04 80 0a ...
```

Has Flag: 0x01 (for the entire row)

Flag: 0x00.

FlaggedPropertyValue: 0x321ff10000000001. This property is **PidTagFolderId** (0x67480014) ([MS-OXCFOLD] section 2.2.2.2.1.6).

The format follows this pattern as covered in section 4.4.2, the server response buffer for the first **RopQueryRows** example.

Because this example is for messages with categories, there is an interesting case when one message has multiple categories assigned to it. Further into the buffer are the following sets of properties.

Property tag	Property value
0x674D0014 (**PidTagInstID** (section 2.2.1.1))	0xb773f10000000001
0x674E0003 (**PidTagInstanceNum** (section 2.2.1.2))	1
0x8008001F (**PidTagAddressBookIsMemberOfDistributionList** ([MS-OXOABK] section 4.5.3.2))	Category1

Property tag	Property value
0x674D0014 (**PidTagInstID**)	0xb773f10000000001
0x674E0003 (**PidTagInstanceNum**)	2
0x8008001F (**PidTagAddressBookIsMemberOfDistributionList**)	Category2

The same message appears twice in the contents table due to the category grouping. The **PidTagInstanceNum** property makes this phenomenon easily recognizable.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

- Microsoft Exchange Server 2003
- Microsoft Exchange Server 2007
- Microsoft Exchange Server 2010
- Microsoft Exchange Server 2013
- Microsoft Exchange Server 2016
- Microsoft Office Outlook 2003
- Microsoft Office Outlook 2007
- Microsoft Outlook 2010
- Microsoft Outlook 2013
- Microsoft Outlook 2016

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

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

<1> Section 2.2.2.1.3: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<2> Section 2.2.2.1.4: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<3> Section 2.2.2.2.1: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<4> Section 2.2.2.2.2: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<5> Section 2.2.2.3.1: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<6> Section 2.2.2.3.2: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<7> Section 2.2.2.4.1: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<8> Section 2.2.2.4.2: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<9> Section 2.2.2.5.1: Office Outlook 2003, Office Outlook 2007, Outlook 2010, and the initial release version of Outlook 2013 do not support the **Execute** request type. The **Execute** request type was introduced in Microsoft Outlook 2013 Service Pack 1 (SP1).

<10> Section 2.2.2.6: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<11> Section 2.2.2.6.2: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<12> Section 2.2.2.7: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<13> Section 2.2.2.7.2: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<14> Section 2.2.2.11: Exchange 2010 does not support the **RopSeekRowFractional** ROP, but Microsoft Exchange Server 2010 Service Pack 1 (SP1), Exchange 2013, and Exchange 2016 do support this ROP.

<15> Section 2.2.2.14.2: Exchange 2007 and Exchange 2010 always set this value to "FALSE" (0x00) for hierarchy tables.

<16> Section 2.2.2.17.1: Exchange 2013 and Exchange 2016 do not support a value greater than 0 for the **MaxRowCount** field.

<17> Section 3.1.4.1.1: Office Outlook 2007 never performs asynchronous table ROPs against the server. Exchange 2010, Exchange 2013, and Exchange 2016 never honor requests to perform asynchronous table ROPs.

<18> Section 3.1.4.3: Exchange 2010 does not support the **RopSeekRowFractional** ROP, but Exchange 2010 SP1, Exchange 2013, and Exchange 2016 do support this ROP.

<19> Section 3.1.5: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<20> Section 3.2.5.1: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<21> Section 3.2.5.2: Exchange 2003 and Exchange 2007 do not fail the ROP if the columns are not set, but use a set of columns that are undefined.

<22> Section 3.2.5.2: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<23> Section 3.2.5.3: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<24> Section 3.2.5.4: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<25> Section 3.2.5.5: Exchange 2010, Exchange 2013, and Exchange 2016 set the **Origin** field to "BOOKMARK_END".

<26> Section 3.2.5.7: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<27> Section 3.2.5.9: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<28> Section 3.2.5.10: Exchange 2007 returns ecSuccess when the bookmark (2) has become invalid because of a **RopSortTable** or **RopResetTable** ROP request. If the bookmark has become invalid because of a **RopRestrict** ROP request on the content table, Exchange 2007 returns ecNotFound.

<29> Section 3.2.5.11: Exchange 2010 does not support the **RopSeekRowFractional** ROP, but Exchange 2010 SP1, Exchange 2013, and Exchange 2016 do support this ROP.

<30> Section 3.2.5.11: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<31> Section 3.2.5.12: Exchange 2003 and Exchange 2007 do not require that a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1) be sent before sending a **RopCreateBookmark** ROP request ([MS-OXCROPS] section 2.2.5.11).

<32> Section 3.2.5.12: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<33> Section 3.2.5.14: Exchange 2007 returns ecSuccess when the bookmark has become invalid because of a **RopSortTable** or **RopResetTable** ROP request. If the bookmark has become invalid because of a **RopRestrict** ROP request on the content table, Exchange 2007 returns ecNotFound.

<34> Section 3.2.5.14: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

<35> Section 3.2.5.15: Exchange 2003 and Exchange 2007 do not conform to the specification and currently send the value "ecNotSupported" in the **ReturnValue** field of the ROP response buffer for contents tables.

<36> Section 3.2.5.16: Exchange 2010, Exchange 2013, and Exchange 2016 do not support asynchronous operations on tables and ignore the TABL_ASYNC flags, as described in section 2.2.2.1.4.

7 Change Tracking

This section identifies changes that were made to this document since the last release. 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.
- The removal of a document from the documentation set.

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 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 changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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.
- 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 dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
3.2.5.11 Processing RopSeekRowFractional	Revised the product behavior note to more clearly distinguish version support for this ROP.	N	Product behavior note updated.
3.2.5.15 Processing RopFreeBookmark	Updated ecNullObject error code description and added ecInvalidBookmark error code to table.	Y	New content added.
6 Appendix A: Product Behavior	Updated the list of applicable products.	Y	Content update.

8 Index

P

Q

R

S

T

V

W