

[MS-OXCTABL]: Table Object Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Revised and edited technical content.
12/03/2008	1.03		Revised and edited technical content.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	3.1.0	Minor	Updated the technical content.
02/10/2010	3.2.0	Minor	Updated the technical content.
05/05/2010	4.0.0	Major	Updated and revised the technical content.
08/04/2010	5.0	Major	Significantly changed the technical content.
11/03/2010	6.0	Major	Significantly changed the technical content.
03/18/2011	7.0	Major	Significantly changed the technical content.
08/05/2011	7.1	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction	8
1.1 Glossary	8
1.2 References	9
1.2.1 Normative References	9
1.2.2 Informative References	9
1.3 Overview	10
1.3.1 Table Notifications	10
1.4 Relationship to Other Protocols	10
1.5 Prerequisites/Preconditions	11
1.6 Applicability Statement	11
1.7 Versioning and Capability Negotiation	11
1.8 Vendor-Extensible Fields	11
1.9 Standards Assignments	11
2 Messages	12
2.1 Transport	12
2.2 Message Syntax	12
2.2.1 Table-Specific Properties	12
2.2.1.1 PidTagInstID	12
2.2.1.2 PidTagInstanceNum	12
2.2.1.3 PidTagRowType	12
2.2.1.4 PidTagDepth	13
2.2.1.5 PidTagContentCount	13
2.2.1.6 PidTagContentUnreadCount	13
2.2.2 Table ROPs	13
2.2.2.1 Table ROP Constants	13
2.2.2.1.1 Predefined Bookmarks	13
2.2.2.1.2 Custom Bookmarks	14
2.2.2.1.3 TableStatus	14
2.2.2.1.4 Asynchronous Flags	14
2.2.2.2 RopSetColumns Semantics	14
2.2.2.2.1 Request Field Overview	15
2.2.2.2.1.1 SetColumnsFlags	15
2.2.2.2.1.2 PropertyTagCount	15
2.2.2.2.1.3 PropertyTags	15
2.2.2.2.2 Response Field Overview	15
2.2.2.2.2.1 TableStatus	15
2.2.2.3 RopSortTable Semantics	15
2.2.2.3.1 Request Field Overview	15
2.2.2.3.1.1 SortTableFlags	15
2.2.2.3.1.2 SortOrderCount	16
2.2.2.3.1.3 CategoryCount	16
2.2.2.3.1.4 ExpandedCount	16
2.2.2.3.1.5 SortOrders	16
2.2.2.3.2 Response Field Overview	17
2.2.2.3.2.1 TableStatus	17
2.2.2.4 RopRestrict Semantics	17
2.2.2.4.1 Request Field Overview	17
2.2.2.4.1.1 RestrictFlags	17
2.2.2.4.1.2 RestrictionDataSize	17

2.2.2.4.1.3	RestrictionData	17
2.2.2.4.2	Response Field Overview	17
2.2.2.4.2.1	TableStatus	17
2.2.2.5	RopQueryRows Semantics	18
2.2.2.5.1	Request Field Overview	18
2.2.2.5.1.1	QueryRowsFlags	18
2.2.2.5.1.2	ForwardRead	18
2.2.2.5.1.3	RowCount	18
2.2.2.5.2	Response Field Overview	18
2.2.2.5.2.1	Origin	18
2.2.2.5.2.2	RowCount	19
2.2.2.5.2.3	RowData	19
2.2.2.6	RopAbort Semantics	19
2.2.2.6.1	Request Field Overview	19
2.2.2.6.2	Response Field Overview	19
2.2.2.6.2.1	TableStatus	19
2.2.2.7	RopGetStatus Semantics	19
2.2.2.7.1	Request Field Overview	19
2.2.2.7.2	Response Field Overview	20
2.2.2.7.2.1	TableStatus	20
2.2.2.8	RopQueryPosition Semantics	20
2.2.2.8.1	Request Field Overview	20
2.2.2.8.2	Response Field Overview	20
2.2.2.8.2.1	Numerator	20
2.2.2.8.2.2	Denominator	20
2.2.2.9	RopSeekRow Semantics	20
2.2.2.9.1	Request Field Overview	20
2.2.2.9.1.1	Origin	20
2.2.2.9.1.2	RowCount	20
2.2.2.9.1.3	WantRowMovedCount	21
2.2.2.9.2	Response Field Overview	21
2.2.2.9.2.1	HasSoughtLess	21
2.2.2.9.2.2	RowsSought	21
2.2.2.10	RopSeekRowBookmark Semantics	21
2.2.2.10.1	Request Field Overview	22
2.2.2.10.1.1	BookmarkSize	22
2.2.2.10.1.2	Bookmark	22
2.2.2.10.1.3	RowCount	22
2.2.2.10.1.4	WantRowMovedCount	22
2.2.2.10.2	Response Field Overview	22
2.2.2.10.2.1	RowNoLongerVisible	22
2.2.2.10.2.2	HasSoughtLess	23
2.2.2.10.2.3	RowsSought	23
2.2.2.11	RopSeekRowFractional Semantics	23
2.2.2.11.1	Request Field Overview	23
2.2.2.11.1.1	Numerator	23
2.2.2.11.1.2	Denominator	23
2.2.2.11.2	Response Field Overview	23
2.2.2.12	RopCreateBookmark Semantics	24
2.2.2.12.1	Request Field Overview	24
2.2.2.12.2	Response Field Overview	24
2.2.2.12.2.1	BookmarkSize	24
2.2.2.12.2.2	Bookmark	24

2.2.2.13	RopQueryColumnsAll Semantics.....	24
2.2.2.13.1	Request Field Overview	24
2.2.2.13.2	Response Field Overview	24
2.2.2.13.2.1	PropertyTagCount	24
2.2.2.13.2.2	PropertyTags.....	24
2.2.2.14	RopFindRow Semantics.....	24
2.2.2.14.1	Request Field Overview	25
2.2.2.14.1.1	FindRowFlags	25
2.2.2.14.1.2	RestrictionDataSize	25
2.2.2.14.1.3	RestrictionData.....	25
2.2.2.14.1.4	Origin.....	25
2.2.2.14.1.5	BookmarkSize	25
2.2.2.14.1.6	Bookmark	25
2.2.2.14.2	Response Field Overview	26
2.2.2.14.2.1	RowNoLongerVisible.....	26
2.2.2.14.2.2	HasRowData	26
2.2.2.14.2.3	RowData.....	26
2.2.2.15	RopFreeBookmark Semantics	26
2.2.2.15.1	Request Field Overview	26
2.2.2.15.1.1	BookmarkSize	26
2.2.2.15.1.2	Bookmark.....	26
2.2.2.15.2	Response Field Overview	27
2.2.2.16	RopResetTable Semantics	27
2.2.2.16.1	Request Field Overview	27
2.2.2.16.2	Response Field Overview	27
2.2.2.17	RopExpandRow Semantics	27
2.2.2.17.1	Request Field Overview	27
2.2.2.17.1.1	MaxRowCount	27
2.2.2.17.1.2	CategoryId.....	27
2.2.2.17.2	Response Field Overview	28
2.2.2.17.2.1	ExpandedRowCount	28
2.2.2.17.2.2	RowCount.....	28
2.2.2.17.2.3	RowData.....	28
2.2.2.18	RopCollapseRow Semantics.....	28
2.2.2.18.1	Request Field Overview	28
2.2.2.18.1.1	CategoryId.....	28
2.2.2.18.2	Response Field Overview	28
2.2.2.18.2.1	CollapsedRowCount.....	28
2.2.2.19	RopGetCollapseState Semantics	28
2.2.2.19.1	Request Field Overview	29
2.2.2.19.1.1	RowId	29
2.2.2.19.1.2	RowIndex.....	29
2.2.2.19.2	Response Field Overview	29
2.2.2.19.2.1	CollapseStateSize	29
2.2.2.19.2.2	CollapseState	29
2.2.2.20	RopSetCollapseState Semantics.....	29
2.2.2.20.1	Request Field Overview	29
2.2.2.20.1.1	CollapseStateSize	29
2.2.2.20.1.2	CollapseState	30
2.2.2.20.2	Response Field Overview	30
2.2.2.20.2.1	BookmarkSize	30
2.2.2.20.2.2	Bookmark	30

3	Protocol Details	31
3.1	Client Details	31
3.1.1	Abstract Data Model	31
3.1.2	Timers	31
3.1.3	Initialization	31
3.1.4	Higher-Layer Triggered Events	31
3.1.4.1	Preparing the Table	31
3.1.4.1.1	Asynchronous Table Preparation	32
3.1.4.2	Querying the Table	33
3.1.4.3	Advancing the Table	33
3.1.4.4	Getting Table State	34
3.1.4.5	Registering For Notifications	34
3.1.5	Message Processing Events and Sequencing Rules	34
3.1.5.1	Processing Notifications	35
3.1.6	Timer Events	35
3.1.7	Other Local Events	35
3.2	Server Details	35
3.2.1	Abstract Data Model	35
3.2.2	Timers	35
3.2.3	Initialization	35
3.2.4	Higher-Layer Triggered Events	35
3.2.5	Message Processing Events and Sequencing Rules	35
3.2.5.1	Processing Asynchronous Requests	35
3.2.5.2	Processing RopSetColumns	36
3.2.5.3	Processing RopSortTable	37
3.2.5.4	Processing RopRestrict	37
3.2.5.5	Processing RopQueryRows	38
3.2.5.6	Processing RopAbort	39
3.2.5.7	Processing RopGetStatus	39
3.2.5.8	Processing RopQueryPosition	40
3.2.5.9	Processing RopSeekRow	40
3.2.5.10	Processing RopSeekRowBookmark	40
3.2.5.11	Processing RopSeekRowFractional	41
3.2.5.12	Processing RopCreateBookmark	42
3.2.5.13	Processing RopQueryColumnsAll	42
3.2.5.14	Processing RopFindRow	42
3.2.5.15	Processing RopFreeBookmark	43
3.2.5.16	Processing RopResetTable	44
3.2.5.17	Processing RopExpandRow	44
3.2.5.18	Processing RopCollapseRow	45
3.2.5.19	Processing RopGetCollapseState	45
3.2.5.20	Processing RopSetCollapseState	45
3.2.6	Timer Events	46
3.2.7	Other Local Events	46
4	Protocol Examples	47
4.1	Obtaining a Message List	47
4.1.1	Client Request Buffer	47
4.1.2	Server Response to Client Request	48
4.2	Setting the Columns on a Table	48
4.2.1	Client Request Buffer	49
4.2.2	Server Response to Client Request	50
4.3	Sorting a Table by Time Delivered	50

4.3.1	Client Request Buffer	50
4.3.2	Server Response to Client Request	51
4.4	Querying Rows	51
4.4.1	Client Request Buffer	51
4.4.2	Server Response to Client Request	52
4.5	Working with Categories	53
4.5.1	Sorting a Table by Category	53
4.5.1.1	Client Request Buffer	53
4.5.1.2	Server Response to Client Request	54
4.5.2	Expanding a Category Row	55
4.5.2.1	Client Request Buffer	55
4.5.2.2	Server Response to Client Request	56
4.5.3	Querying Rows with Category View.....	56
4.5.3.1	Client Request Buffer	56
4.5.3.2	Server Response to Client Request	57
5	Security.....	59
5.1	Security Considerations for Implementers.....	59
5.2	Index of Security Parameters	59
6	Appendix A: Product Behavior	60
7	Change Tracking.....	63
8	Index	65

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

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

flags
handle
little-endian
remote procedure call (RPC)

The following terms are defined in [\[MS-OXGLOS\]](#):

attachments table
binary large object (BLOB)
bookmark
category
contents table
Folder object
handle array
header row
hierarchy table
instance
multivalue property
permission
property ID
property tag
property type
remote operation (ROP)
restriction
ROP request
ROP request buffer
ROP response
ROP response buffer
rule
rules table
search criteria
sort order
Table object

The following terms are specific to this document:

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

leaf row: A row that is in a category.

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

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

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

1.2 References

1.2.1 Normative References

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

[MS-DTYP] Microsoft Corporation, "[Windows Data Types](#)".

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

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

[MS-OXCNOTIF] Microsoft Corporation, "[Core Notifications Protocol Specification](#)".

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

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

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

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

1.2.2 Informative References

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

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

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

[MS-OXCPerm] Microsoft Corporation, "[Exchange Access and Operation Permissions Protocol Specification](#)".

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

[MS-OXOABK] Microsoft Corporation, "[Address Book Object Protocol Specification](#)".

[MS-OXOMSG] Microsoft Corporation, "[E-Mail Object Protocol Specification](#)".

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 (2)**, 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 (2)** 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 (2) represents their **category (5)**. A category (5) can be nested inside another category (5). The client can retrieve a **binary large object (BLOB)** that specifies which categories (5) 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 (5) that are based on multivalue properties will display the multivalue instances under each header row (2) representing a value that is set on that row. The row that is displayed under a given header row (2) will include the single property value specified by the header row (2), 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 E-Mail 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.

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** - [\[MS-OXCFOOLD\]](#) section [2.2.14](#)
- Hierarchy table - [\[MS-OXCFOOLD\]](#) section [2.2.13](#)
- **Attachments table** - [\[MS-OXCMSG\]](#) section 2.2.3.17
- **Permissions table** - [\[MS-OXCPerm\]](#) section 2.2.1.1
- 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 the results from the **EcDoConnectEx** method, as described in [\[MS-OXCRPC\]](#) section 3.1.4.11.

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 eight (8), as described in [\[MS-OXCRPC\]](#) section 3.1.9.2.

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 from, respectively, the server by using the underlying Wire Format Protocol, as specified in [\[MS-OXCRPC\]](#).

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 (5). 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 2.805) 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 2.804) 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 2.996) 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 (2) with no rows inside the category (5).
TBL_EXPANDED_CATEGORY	0x00000003	The row is a header row (2) that is expanded.

Row name	Value	Meaning
TBL_COLLAPSED_CATEGORY	0x00000004	The row is a header row (2) that is collapsed.

2.2.1.4 PidTagDepth

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

The **PidTagDepth** property ([\[MS-OXPROPS\]](#) section 2.734) specifies the number of nested categories (5) in which a given row is contained. For example, if a row is contained within two header rows (2), 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 (2). This property is set whether the header row (2) 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 (2) that have the **PidTagRead** property ([\[MS-OXPROPS\]](#) section 2.936) set to "FALSE" (0x00). This value is set whether the header row (2) 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 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 Semantics

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.

2.2.2.2.1 Request Field Overview

2.2.2.2.1.1 SetColumnsFlags

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **SetColumnsFlags** field contains an OR'ed combination of the asynchronous flags, which are specified in section [2.2.2.1.4.<3>](#). The field MUST NOT have any of the other bits set.

2.2.2.2.1.2 PropertyTagCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **PropertyTagCount** field specifies the number of **property tags** in the **PropertyTags** field. This value MUST be 1 or greater.

2.2.2.2.1.3 PropertyTags

The **PropertyTags** field contains an array of **PropertyTag** structures, as described in [\[MS-OXCADATA\]](#) 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 multivalued property, and the client wants multivalued instances based on this property, it MUST also set the **MultivalueInstance** bit (0x2000), as specified in [\[MS-OXCADATA\]](#) section 2.11.1.2, 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 Response Field Overview

2.2.2.2.2.1 TableStatus

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **TableStatus** field 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 [2.2.2.1.3](#).

2.2.2.3 RopSortTable Semantics

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.

2.2.2.3.1 Request Field Overview

2.2.2.3.1.1 SortTableFlags

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **SortTableFlags** field contains an OR'ed combination of the asynchronous flags, which are specified in section [2.2.2.1.4.<5>](#). The field MUST NOT have any of the other bits set.

2.2.2.3.1.2 SortOrderCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **SortOrderCount** field specifies the number of **SortOrder** structures, as specified in [\[MS-OXCADATA\]](#) section 2.13.1, in the **SortOrders** field, as specified in section [2.2.2.3.1.5](#).

2.2.2.3.1.3 CategoryCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **CategoryCount** field specifies the number of **SortOrder** structures, as specified in [\[MS-OXCADATA\]](#) section 2.13.1, in the **SortOrders** field, as specified in section [2.2.2.3.1.5](#), 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.

2.2.2.3.1.4 ExpandedCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **ExpandedCount** field specifies the number of categories (5) that start in the expanded state. This value MUST be in the range 0 to the value of the **CategoryCount** field. The first categories (5) 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 (5) are expanded.

2.2.2.3.1.5 SortOrders

The **SortOrders** field contains an array of **SortOrder** structures, as specified in [\[MS-OXCADATA\]](#) 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 (5). When the value of the **SortOrderCount** field exceeds the value of the **CategoryCount** field, indicating that there are more sort keys than categories (5), categories (5) 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 (5).

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 [\[MS-OXCADATA\]](#) section 2.11.1.2, 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 (5) in the table will be not be sorted by the column specified in the category sort. The categories (5) will be sorted according to each category's (5) maximum value of the specified column. Any **SortOrder** structures after this one will subsort the rows within each category (5).

2.2.2.3.2 Response Field Overview

2.2.2.3.2.1 TableStatus

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **TableStatus** field indicates the status of asynchronous operations being performed on the table. [<6>](#) It MUST have one of the table status values that are specified in section [2.2.2.1.3](#).

2.2.2.4 RopRestrict Semantics

The **RopRestrict** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.3) establishes a restriction (2) on a table. Applying a restriction (2) 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 (2). This ROP is valid only on Table objects.

2.2.2.4.1 Request Field Overview

2.2.2.4.1.1 RestrictFlags

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **RestrictFlags** field contains an OR'ed combination of the asynchronous flags, as specified in section [2.2.2.1.4.<7>](#) The field MUST NOT have any of the other bits set.

2.2.2.4.1.2 RestrictionDataSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **RestrictionDataSize** field specifies the size, in bytes, of the **RestrictionData** field.

2.2.2.4.1.3 RestrictionData

The **RestrictionData** field is a restriction (2) that is applied to the table. For details about restrictions (2), see [\[MS-OXCADATA\]](#) section 2.12. It has a number of bytes equal to the value of the **RestrictionDataSize** field.

2.2.2.4.2 Response Field Overview

2.2.2.4.2.1 TableStatus

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **TableStatus** field indicates the status of asynchronous operations being performed on the table. [<8>](#) It MUST have one of the table status values that are specified in section [2.2.2.1.3](#).

2.2.2.5 RopQueryRows Semantics

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.

2.2.2.5.1 Request Field Overview

2.2.2.5.1.1 QueryRowsFlags

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **QueryRowsFlags** field 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 the <i>pulFlags</i> parameter of the EcDoRpcExt2 method, as specified in [MS-OXCRPC] section 3.1.4.12. 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 contain both the **Advance** and **NoAdvance** flags set simultaneously. The field MUST NOT have any of the other bits set.

2.2.2.5.1.2 ForwardRead

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **ForwardRead** field specifies 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.

2.2.2.5.1.3 RowCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **RowCount** field specifies the maximum number of rows to be returned.

2.2.2.5.2 Response Field Overview

2.2.2.5.2.1 Origin

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **Origin** field identifies the cursor position. It MUST be set to one of the predefined **bookmark (2)** values specified in section [2.2.2.1.1](#).

2.2.2.5.2.2 RowCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **RowCount** field 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.

2.2.2.5.2.3 RowData

The **RowData** field 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 **RowSetColumns** 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 **RowRestrict** ROP request ([\[MS-OXCROPS\]](#) section 2.2.5.3). If the **RowRestrict** 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 **RowSortTable** ROP request ([\[MS-OXCROPS\]](#) section 2.2.5.2). If the **RowSortTable** 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 RowAbort Semantics

The **RowAbort** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.5) attempts to stop any asynchronous table operations that are currently in progress. [<9>](#) This ROP is valid only on Table objects.

2.2.2.6.1 Request Field Overview

There are no ROP-specific request fields.

2.2.2.6.2 Response Field Overview

2.2.2.6.2.1 TableStatus

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **TableStatus** field indicates the status of asynchronous operations being performed on the table before the abort. [<10>](#) Its value **MUST** be one of the table status values that are specified in [2.2.2.1.3](#).

2.2.2.7 RowGetStatus Semantics

The **RowGetStatus** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.6) retrieves information about the current status of asynchronous operations being performed on the table. [<11>](#) This ROP is valid only on Table objects.

2.2.2.7.1 Request Field Overview

There are no ROP-specific request fields.

2.2.2.7.2 Response Field Overview

2.2.2.7.2.1 TableStatus

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **TableStatus** field indicates the status of asynchronous operations being performed on the table. [<12>](#) Its value MUST be one of the table status values that are specified in [2.2.2.1.3](#).

2.2.2.8 RopQueryPosition Semantics

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.

2.2.2.8.1 Request Field Overview

There are no ROP-specific request fields.

2.2.2.8.2 Response Field Overview

2.2.2.8.2.1 Numerator

Data type: **ULONG** ([\[MS-DTYP\]](#))

The **Numerator** field contains the index (0-based) of the current row. Its value MUST be greater than or equal to 0x00000000.

2.2.2.8.2.2 Denominator

Data type: **ULONG** ([\[MS-DTYP\]](#))

The **Denominator** field contains 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 Semantics

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 (2), as specified in section [2.2.2.1.1](#), and the number of rows to move (forward or backwards) from that bookmark (2). This ROP is valid only on Table objects.

2.2.2.9.1 Request Field Overview

2.2.2.9.1.1 Origin

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **Origin** field contains the **Bookmark** field array indicating the starting position of the seek operation. This field's value MUST be one of the predefined bookmark (2) values that are specified in section [2.2.2.1.1](#).

2.2.2.9.1.2 RowCount

Data type: **LONG** ([\[MS-DTYP\]](#))

The **RowCount** field contains the number of rows to seek, starting from the bookmark (2). To seek forward from the bookmark (2), the value MUST be positive; to seek backwards, the value MUST be negative.

2.2.2.9.1.3 WantRowMovedCount

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **WantRowMovedCount** field 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** (section [2.2.2.9.2.1](#)) and **RowsSought** (section [2.2.2.9.2.2](#)) fields 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 Response Field Overview

2.2.2.9.2.1 HasSoughtLess

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **HasSoughtLess** field 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".

2.2.2.9.2.2 RowsSought

Data type: **LONG** ([\[MS-DTYP\]](#))

The **RowsSought** field 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 Semantics

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 (2), as specified in section [2.2.2.1.2](#), and the number of rows to move (forward or backwards) from that bookmark (2).

This ROP is distinguished from the **RopSeekRow** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.8) in that the bookmark (2) 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.

2.2.2.10.1 Request Field Overview

2.2.2.10.1.1 BookmarkSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **BookmarkSize** field specifies the size, in bytes, of the **Bookmark** field.

2.2.2.10.1.2 Bookmark

Data type: Array of bytes

The **Bookmark** field is a bookmark (2) 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 (2) MUST NOT have been previously freed by using the **RopFreeBookmark** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.14).

2.2.2.10.1.3 RowCount

Data type: **LONG** ([\[MS-DTYP\]](#))

The **RowCount** field specifies the number of rows to seek, starting from the bookmark (2). To seek forward from the bookmark (2), the value MUST be positive; to seek backward, the value MUST be negative.

2.2.2.10.1.4 WantRowMovedCount

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **WantRowMovedCount** field specifies whether the actual number of rows moved is 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** (section [2.2.2.9.2.1](#)) and **RowsSought** (section [2.2.2.9.2.2](#)) fields 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 Response Field Overview

2.2.2.10.2.1 RowNoLongerVisible

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **RowNoLongerVisible** field indicates whether the row to which the bookmark (2) pointed is no longer visible. This field MUST be set to "TRUE" (0x01) if the row to which the bookmark (2) pointed has been removed from the table. (For example, the row's properties changed so that they didn't match the restriction (2), the row was deleted, or the row's header row (2) has been collapsed.) Otherwise, this field MUST be set to "FALSE" (0x00).

When the row to which the bookmark (2) pointed is no longer visible, the bookmark (2) will point to the next row in the table. In this case, the seek will begin from the next row after the bookmark (2) in the table.

2.2.2.10.2.2 HasSoughtLess

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **HasSoughtLess** field 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".

2.2.2.10.2.3 RowsSought

Data type: **LONG** ([\[MS-DTYP\]](#))

The **RowsSought** field 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.

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" (0x01) and **MUST** be ignored if **WantRowMovedCount** is set to "FALSE" (0x00).

2.2.2.11 RopSeekRowFractional Semantics

The **RopSeekRowFractional** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.10) moves the table cursor to an approximate position in the table. [<13>](#) The new location is specified as a fraction of the table size. This ROP is valid only on Table objects.

2.2.2.11.1 Request Field Overview

2.2.2.11.1.1 Numerator

Data type: **ULONG** ([\[MS-DTYP\]](#))

The **Numerator** field is 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.

2.2.2.11.1.2 Denominator

Data type: **ULONG** ([\[MS-DTYP\]](#))

The **Denominator** field is the denominator of the fractional position. The value **MUST NOT** be set to zero (0x00000000).

2.2.2.11.2 Response Field Overview

There are no ROP-specific response fields.

2.2.2.12 RopCreateBookmark Semantics

The **RopCreateBookmark** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.11) creates a new bookmark (2) at the current cursor position in the table. This ROP is valid only on Table objects.

2.2.2.12.1 Request Field Overview

There are no ROP-specific request fields.

2.2.2.12.2 Response Field Overview

2.2.2.12.2.1 BookmarkSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **BookmarkSize** field specifies the size, in bytes, of the **Bookmark** field. This field **MUST** be present.

2.2.2.12.2.2 Bookmark

The **Bookmark** field contains the bookmark (2) 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 Semantics

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.

2.2.2.13.1 Request Field Overview

There are no ROP-specific request fields.

2.2.2.13.2 Response Field Overview

2.2.2.13.2.1 PropertyTagCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **PropertyTagCount** field specifies the number of property tags in the **PropertyTags** field.

2.2.2.13.2.2 PropertyTags

The **PropertyTags** field contains 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 [\[MS-OXCADATA\]](#) section 2.9.

2.2.2.14 RopFindRow Semantics

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 (2). This ROP is valid only on Table objects.

2.2.2.14.1 Request Field Overview

2.2.2.14.1.1 FindRowFlags

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **FindRowFlags** field contains an OR'ed combination of any of the following bit flags.

Value	Meaning
0x00	Perform the find forwards.
0x01	Perform the find backwards.

The field **MUST NOT** have any of the other bits set.

2.2.2.14.1.2 RestrictionDataSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **RestrictionDataSize** field specifies the size, in bytes, of the **RestrictionData** field.

2.2.2.14.1.3 RestrictionData

The **RestrictionData** field contains a restriction (2) that specifies the criteria to be used for the search. For details about restrictions (2), see [\[MS-OXCDATA\]](#) section 2.12.

2.2.2.14.1.4 Origin

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **Origin** field specifies the bookmark (2) and **MUST** be set either to one of the predefined bookmark (2) values specified in section [2.2.2.1.1](#) or to **BOOKMARK_CUSTOM**, as specified in section [2.2.2.1.2](#).

2.2.2.14.1.5 BookmarkSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **BookmarkSize** field specifies the size, in bytes, of the **Bookmark** field. If the **Origin** field specifies a predefined bookmark, as specified in section [2.2.2.1.1](#), this field **MUST** be set to zero (0x0000).

2.2.2.14.1.6 Bookmark

The **Bookmark** field specifies the bookmark (2) from which to begin the search. The bookmark (2) **MUST** be one that was returned by a previous **RopCreateBookmark** ROP request ([\[MS-OXCROPS\]](#) section 2.2.5.11). The bookmark (2) **MUST NOT** have been previously freed using the **RopFreeBookmark** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.14). If the **Origin** field specifies a predefined bookmark (2), as specified in section [2.2.2.1.1](#), this field **MUST NOT** be present in the request.

2.2.2.14.2 Response Field Overview

2.2.2.14.2.1 RowNoLongerVisible

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **RowNoLongerVisible** field indicates whether the row to which the bookmark (2) pointed is no longer visible. This field SHOULD [<14>](#) be set to "TRUE" (0x01) if the row to which the bookmark (2) pointed has been removed from the table. (For example, the row's properties changed so that it didn't match the restriction (2), the row was deleted, or the row's header row (2) has been collapsed.) Otherwise, this field MUST be set to "FALSE" (0x00).

When the row to which the bookmark (2) pointed is no longer visible, the search will begin from the next row after the bookmark (2) in the table. When searching backward, the search will not consider the row to which the bookmark (2) is currently pointing, but will begin at the previous row.

2.2.2.14.2.2 HasRowData

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **HasRowData** field specifies whether a row meeting the specified search criteria was found. If a row that meets the specified search criteria was found, this field MUST be set to "TRUE" (0x01).

2.2.2.14.2.3 RowData

The **RowData** field contains 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 Semantics

The **RopFreeBookmark** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.14) frees the memory associated with a bookmark (2) that was returned by a previous **RopCreateBookmark** ROP request ([\[MS-OXCROPS\]](#) section 2.2.5.11). After the bookmark (2) has been released, attempts to use the bookmark (2) will fail with "ecInvalidBookmark". This ROP is valid only on Table objects.

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](#).

2.2.2.15.1 Request Field Overview

2.2.2.15.1.1 BookmarkSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **BookmarkSize** field specifies the size, in bytes, of the **Bookmark** field.

2.2.2.15.1.2 Bookmark

The **Bookmark** field specifies the bookmark (2) to be freed. The bookmark (2) MUST be one that was returned by a previous **RopCreateBookmark** ROP request ([\[MS-OXCROPS\]](#) section 2.2.5.11). The bookmark (2) MUST NOT have been previously freed using the **RopFreeBookmark** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.14).

2.2.2.15.2 Response Field Overview

There are no ROP-specific response fields.

2.2.2.16 RopResetTable Semantics

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

- Removes the existing column set, restriction (2), and sort order from the table.
- Invalidates bookmarks (2).
- 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 (2) 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).

2.2.2.16.1 Request Field Overview

There are no ROP-specific request fields.

2.2.2.16.2 Response Field Overview

There are no ROP-specific response fields.

2.2.2.17 RopExpandRow Semantics

The **RopExpandRow** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.16) expands a collapsed category (5) of a table and returns the rows that belong in the newly expanded category (5). The maximum number of **leaf rows** to be returned can be specified. This ROP is valid only on Table objects.

2.2.2.17.1 Request Field Overview

2.2.2.17.1.1 MaxRowCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **MaxRowCount** field specifies the maximum number of leaf rows to be returned in the response.

2.2.2.17.1.2 CategoryId

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **CategoryId** field specifies the category (5) to be expanded. This field is set to the value of the **PidTagInstID** property (section [2.2.1.1](#)) of the category's (5) header row (2).

2.2.2.17.2 Response Field Overview

2.2.2.17.2.1 ExpandedRowCount

Data type: **ULONG** ([\[MS-DTYP\]](#))

The **ExpandedRowCount** field specifies the total number of rows that are in the expanded category (5).

2.2.2.17.2.2 RowCount

Data type: **WORD** ([\[MS-DTYP\]](#))

The **RowCount** field 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.
- The value of the **ExpandedRowCount** field in the ROP response buffer.

2.2.2.17.2.3 RowData

The **RowData** field contains an array of **PropertyRow** structures, as specified in [\[MS-OXCADATA\]](#) section 2.8.1, each of which specifies a row in the expanded category (5).

2.2.2.18 RopCollapseRow Semantics

The **RopCollapseRow** ROP ([\[MS-OXCROPS\]](#) section 2.2.5.17) collapses an expanded category (5). This ROP is valid only on Table objects.

2.2.2.18.1 Request Field Overview

2.2.2.18.1.1 CategoryId

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **CategoryId** field specifies the category (5) to be collapsed. This field is set to the value of the **PidTagInstID** property (section [2.2.1.1](#)) of the category's (5) header row (2).

2.2.2.18.2 Response Field Overview

2.2.2.18.2.1 CollapsedRowCount

Data type: **ULONG** ([\[MS-DTYP\]](#))

The **CollapsedRowCount** field specifies the number of rows that have been collapsed.

2.2.2.19 RopGetCollapseState Semantics

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.

2.2.2.19.1 Request Field Overview

2.2.2.19.1.1 RowId

Data type: **BYTE** ([\[MS-DTYP\]](#))

The **RowId** field specifies the row to be preserved as the current cursor, which is returned in the **CollapseState** field of the **RopGetCollapseState ROP response** ([\[MS-OXCROPS\]](#) section 2.2.5.18). 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.

2.2.2.19.1.2 RowInstanceNumber

Data type: **ULONG** ([\[MS-DTYP\]](#))

The **RowInstanceNumber** field 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** ([\[MS-OXCROPS\]](#) section 2.2.5.18).

2.2.2.19.2 Response Field Overview

2.2.2.19.2.1 CollapseStateSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **CollapseStateSize** field specifies the size, in bytes, of the **CollapseState** field.

2.2.2.19.2.2 CollapseState

The **CollapseState** field 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 Semantics

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 (2) 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 (2) returned **MUST** be freed by using the **RopFreeBookmark ROP** ([\[MS-OXCROPS\]](#) section 2.2.5.14).

2.2.2.20.1 Request Field Overview

2.2.2.20.1.1 CollapseStateSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **CollapseStateSize** field specifies the size, in bytes, of the **CollapseState** field.

2.2.2.20.1.2 CollapseState

The **CollapseState** field 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 Response Field Overview

2.2.2.20.2.1 BookmarkSize

Data type: **WORD** ([\[MS-DTYP\]](#))

The **BookmarkSize** field specifies the size, in bytes, of the **Bookmark** field.

2.2.2.20.2.2 Bookmark

The **Bookmark** field contains the bookmark (2) 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 (2). 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 (2). 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 (2). Three predefined bookmarks (2) 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 (2), sort order, restriction (2), 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 (2), 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. <15> 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" 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 (2) 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 (5) into one header row (2), 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 (2), 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 (5), 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<16>** ([\[MS-OXCROPS\]](#) 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 (2) 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 (2), 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 (5), 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 (5) 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 (2) 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 [\[MS-OXCNOTIF\]](#).

3.1.5 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 **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. <17> 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 [≤18>](#) 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-](#)

[\[MS-OXCROPS\]](#) 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 server MUST send the **TableError** notification, as specified in [\[MS-OXCNOTIF\]](#) section 2.2.1.1.1. 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 **TableSortDone**, **TableRestrictionChanged**, or **TableColumnsChanged** 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". 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 either a **RopQueryRows** or a **RopFindRow** ROP is sent before a successful **RopSetColumns** ROP, then the server fails the ROP with "ecNullObject". [<19>](#)

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, [<20>](#) 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 [\[MS-OXCADATA\]](#) 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 [\[MS-OXCADATA\]](#) 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-OXCADATA] section 2.11.1.

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

3.2.5.3 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 (2) 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, <21> 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 (2) to the table, and subsequent requests that operate on the table MUST consider the new restriction (2).

If a restriction (2) is applied to a table, the table MUST appear as if it only contains the rows that match the restriction (2).

When this ROP is sent, the server MUST invalidate all current bookmarks (2) 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 (2) is not specified), then the table MUST be considered as not having any restrictions (2).

If a **RopRestrict** ROP fails, the server SHOULD invalidate the table restriction (2) until a successful **RopRestrict** ROP request is made. The server can restore the previous restriction (2).

If the TBL_ASYNC bit of the **RestrictFlags** field is set, the server MAY [<22>](#) 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.

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 [\[MS-OXCADATA\]](#) 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, the **Origin** field will be set to "BOOKMARK_END" if the **ForwardRead** field in the ROP request is set to "TRUE" or to "BOOKMARK_BEGINNING" if the **ForwardRead** field is set to "FALSE". 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, [<23>](#) 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,	The object on which this ROP was sent is not of type table.

Error code name	Value	Description
	%x02.01.04.80	

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", [<24>](#) 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 name	Value	Description
ecNotSupported	0x80040102, %x02.01.04.80	The object on which this ROP was sent is not of type table.

3.2.5.10 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 (2), as specified in section [2.2.2.1.2](#), as a reference.

If the bookmark (2) 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 MUST set the **ReturnValue** field to "ecInvalidBookmark". If the bookmark (2) 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 (2), or its header row (2) 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-OXCROPS] section 2.4.

Error code name	Value	Description
ecInvalidBookmark	0x80040405, %x05.04.04.80	The bookmark (2) 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. <25>

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", <26> 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-OXCROPS] 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 (2), 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 (2) 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 (2) 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).<27>

The server MUST complete all asynchronous table ROPs prior to executing this ROP or fail the ROP with "ecBusy",<28> 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 [\[MS-OXCADATA\]](#) 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 [\[MS-OXCADATA\]](#) 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. 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 (2), as specified in section 2.2.2.1.2, but the bookmark (2) 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 MUST set the **ReturnValue** field to "ecInvalidBookmark". If the bookmark (2) 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 (2), or its header row (2) 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", <29> 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 (2) (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 (2) 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. <30>

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 (2) after it was released.

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 (2) 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 (2) (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 (2) on the table are invalid.

The server MUST complete all asynchronous table ROPs prior to executing this ROP or fail the ROP with "ecBusy", <31> 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 [\[MS-OXCADATA\]](#) 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 [\[MS-OXCADATA\]](#) 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,	The row specified by the CategoryId field was not collapsed.

Error code name	Value	Description
	%xF8.04.00.00	
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 [\[MS-OXCADATA\]](#) 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 (5) are expanded. It MUST also include a bookmark (2) 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 [\[MS-OXCADATA\]](#) 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 (5) are expanded. It MUST also move the cursor position to the row specified by the bookmark (2).

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 packed with other ROP requests and are compressed and packed in one or more **RPC** calls according to the specification 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

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 [\[MS-OXCADATA\]](#) 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 2.1023))

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** ([\[MS-OXCADATA\]](#) 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 [2.2.14](#).

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 2.2.14).

```
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 2.2.14.2.2, 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 [\[MS-OXCADATA\]](#).

```
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** ([\[MS-OXCFCOLD\]](#) section 2.2.2.2.1.5))

PropertyTag: 0x674a0014 (**PidTagMid** ([\[MS-OXCFCICS\]](#) 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 2.1099))

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 [2.2.2.3.1.5](#).

```
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 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 [2.2.2.3.2.1](#).

```
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 [\[MS-OXCADATA\]](#) 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 2.2.2.2.1.5) (0x67480014) because the order must be maintained.

ErrorType: 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 (5) (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 (5) 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 [2.2.2.3.1.5](#)).

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

PropertyTag: 0x8008301F (**PidTagAddressBookIsMemberOfDistributionList** ([\[MS-OXOABK\]](#) section 2.2.5.3))

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 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 [2.2.2.3.2.1](#).

```
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 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 (5).

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 (5).

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 [\[MS-OXCROPS\]](#) 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 00 f1 88 bd-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 2.2.2.2.1.5).

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 (5), there is an interesting case when one message has multiple categories (5) 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 2.2.5.3))	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® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010

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

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

[<1> Section 2.2.2.1.3:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<2> Section 2.2.2.1.4:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<3> Section 2.2.2.1.1:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<4> Section 2.2.2.2.1:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<5> Section 2.2.2.3.1.1:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<6> Section 2.2.2.3.2.1:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<7> Section 2.2.2.4.1.1:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<8> Section 2.2.2.4.2.1:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<9> Section 2.2.2.6:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

[<10> Section 2.2.2.6.2.1:](#) Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<11> [Section 2.2.2.7](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<12> [Section 2.2.2.7.2.1](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<13> [Section 2.2.2.11](#): Exchange 2010 does not implement the **RopSeekRowFractional** ROP, but Exchange 2010 SP1 does.

<14> [Section 2.2.2.14.2.1](#): Exchange 2007 and Exchange 2010 always set this value to "FALSE" (0x00) for hierarchy tables.

<15> [Section 3.1.4.1.1](#): Office Outlook 2007 never performs asynchronous table ROPs against the server. Exchange 2010 never honors requests to perform asynchronous table ROPs.

<16> [Section 3.1.4.3](#): Exchange 2010 does not support the **RopSeekRowFractional** ROP, but Exchange 2010 SP1 does.

<17> [Section 3.1.5](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<18> [Section 3.2.5.1](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<19> [Section 3.2.5.2](#): Exchange 2003 and Exchange 2007 do not fail this ROP if the columns are not set, but return a set of columns that are undefined.

<20> [Section 3.2.5.2](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<21> [Section 3.2.5.3](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<22> [Section 3.2.5.4](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<23> [Section 3.2.5.7](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<24> [Section 3.2.5.9](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<25> [Section 3.2.5.11](#): Exchange 2010 does not support the **RopSeekRowFractional** ROP, but Exchange 2010 SP1 does.

<26> [Section 3.2.5.11](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<27> [Section 3.2.5.12](#): Exchange 2010 requires that a **RopSetColumns** ROP request be sent, but Exchange 2003 and Exchange 2007 do not require it.

<28> [Section 3.2.5.12](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<29> [Section 3.2.5.14](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

<30> [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.

<31> [Section 3.2.5.16](#): Exchange 2010 does not support asynchronous operations on tables and ignores all values of asynchronous flags.

7 Change Tracking

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
4.5.1.1 Client Request Buffer	Changed property payload to an in-scope property.	N	Content updated.
4.5.3.2 Server Response to Client Request	Changed property payload to an in-scope property.	N	Content updated.

8 Index

A

Abstract data model
[client](#) 31
[server](#) 35
[Applicability](#) 11
[Asynchronous flags](#) 14

C

[Capability negotiation](#) 11
[Change tracking](#) 63
Client
[abstract data model](#) 31
[initialization](#) 31
[message processing](#) 34
[other local events](#) 35
[overview](#) 31
[sequencing rules](#) 34
[timer events](#) 35
[timers](#) 31
Client - higher-layer triggered events
[advancing the table](#) 33
[getting table state](#) 34
[preparing the table](#) 31
[querying the table](#) 33
[registering for notifications](#) 34
Client - message processing
[processing notifications](#) 35
Client - sequencing rules
[processing notifications](#) 35
[Custom bookmarks](#) 14

D

Data model - abstract
[client](#) 31
[server](#) 35

E

Examples
[obtaining a message list](#) 47
[overview](#) 47
[querying rows](#) 51
[setting the columns on a table](#) 48
[sorting a table by time delivered](#) 50
[working with categories](#) 53

F

[Fields - vendor-extensible](#) 11

G

[Glossary](#) 8

H

Higher-layer triggered events
[server](#) 35
Higher-layer triggered events - client
[advancing the table](#) 33
[getting table state](#) 34
[preparing the table](#) 31
[querying the table](#) 33
[registering for notifications](#) 34

I

[Implementer - security considerations](#) 59
[Index of security parameters](#) 59
[Informative references](#) 9
Initialization
[client](#) 31
[server](#) 35
[Introduction](#) 8

M

Message processing
[client](#) 34
message processing - client
[processing notifications](#) 35
message processing - server
[processing asynchronous requests](#) 35
[processing RopAbort](#) 39
[processing RopCollapseRow](#) 45
[processing RopCreateBookmark](#) 42
[processing RopExpandRow](#) 44
[processing RopFindRow](#) 42
[processing RopFreeBookmark](#) 43
[processing RopGetCollapseState](#) 45
[processing RopGetStatus](#) 39
[processing RopQueryColumnsAll](#) 42
[processing RopQueryPosition](#) 40
[processing RopQueryRows](#) 38
[processing RopResetTable](#) 44
[processing RopRestrict](#) 37
[processing RopSeekRow](#) 40
[processing RopSeekRowBookmark](#) 40
[processing RopSeekRowFractional](#) 41
[processing RopSetCollapseState](#) 45
[processing RopSetColumns](#) 36
[processing RopSortTable](#) 37
[Message syntax overview](#) 12
Messages
[message syntax](#) 12
[Table ROPs](#) 13
[Table-Specific Properties](#) 12
[transport](#) 12

N

[Normative references](#) 9

O

Obtaining a message list example
[client request buffer](#) 47
[server response to client request](#) 48
Other local events
[client](#) 35
[server](#) 46
[Overview - message syntax](#) 12
[Overview - table notifications](#) 10
[Overview \(synopsis\)](#) 10

P

[Parameters - security index](#) 59
[PidTagContentCount property](#) 13
[PidTagContentUnreadCount property](#) 13
[PidTagDepth property](#) 13
[PidTagInstanceNum property](#) 12
[PidTagInstID property](#) 12
[PidTagRowType property](#) 12
[Preconditions](#) 11
[Predefined bookmarks](#) 13
[Prerequisites](#) 11
[Product behavior](#) 60

Q

Querying rows example
[client request buffer](#) 51
[server response to client request](#) 52

R

References
[informative](#) 9
[normative](#) 9
[Relationship to other protocols](#) 10
[RopAbort semantics](#) 19
[RopCollapseRow semantics](#) 28
[RopCreateBookmark semantics](#) 24
[RopExpandRow semantics](#) 27
[RopFindRow semantics](#) 24
[RopFreeBookmark semantics](#) 26
[RopGetCollapseState semantics](#) 28
[RopGetStatus semantics](#) 19
[RopQueryColumnsAll semantics](#) 24
[RopQueryPosition semantics](#) 20
[RopQueryRows semantics](#) 18
[RopResetTable semantics](#) 27
[RopRestrict semantics](#) 17
[RopSeekRow semantics](#) 20
[RopSeekRowBookmark semantics](#) 21
[RopSeekRowFractional semantics](#) 23
[RopSetCollapseState semantics](#) 29
[RopSetColumns semantics](#) 14
[RopSortTable semantics](#) 15

S

Security
[implementer considerations](#) 59
[parameter index](#) 59
Sequencing rules

[client](#) 34
Sequencing rules - client
[processing notifications](#) 35
Sequencing rules - server
[processing asynchronous requests](#) 35
[processing RopCollapseRow](#) 45
[processing RopCreateBookmark](#) 42
[processing RopExpandRow](#) 44
[processing RopFindRow](#) 42
[processing RopFreeBookmark](#) 43
[processing RopGetCollapseState](#) 45
[processing RopGetStatus](#) 39
[processing RopQueryColumnsAll](#) 42
[processing RopQueryPosition](#) 40
[processing RopQueryRows](#) 38
[processing RopResetTable](#) 44
[processing RopSeekRow](#) 40
[processing RopSeekRowBookmark](#) 40
[processing RopSeekRowFractional](#) 41
[processing RopSetCollapseState](#) 45
[processing RopSetColumns](#) 36
[processing RopSortTable](#) 37
Server
[abstract data model](#) 35
[higher-layer triggered events](#) 35
[initialization](#) 35
[other local events](#) 46
[timer events](#) 46
[timers](#) 35
Server - message processing
[processing asynchronous requests](#) 35
[processing RopAbort](#) 39
[processing RopCollapseRow](#) 45
[processing RopCreateBookmark](#) 42
[processing RopExpandRow](#) 44
[processing RopFindRow](#) 42
[processing RopFreeBookmark](#) 43
[processing RopGetCollapseState](#) 45
[processing RopGetStatus](#) 39
[processing RopQueryColumnsAll](#) 42
[processing RopQueryPosition](#) 40
[processing RopQueryRows](#) 38
[processing RopResetTable](#) 44
[processing RopRestrict](#) 37
[processing RopSeekRow](#) 40
[processing RopSeekRowBookmark](#) 40
[processing RopSeekRowFractional](#) 41
[processing RopSetCollapseState](#) 45
[processing RopSetColumns](#) 36
[processing RopSortTable](#) 37
Server - sequencing rules
[processing asynchronous requests](#) 35
[processing RopAbort](#) 39
[processing RopCollapseRow](#) 45
[processing RopCreateBookmark](#) 42
[processing RopExpandRow](#) 44
[processing RopFindRow](#) 42
[processing RopFreeBookmark](#) 43
[processing RopGetCollapseState](#) 45
[processing RopGetStatus](#) 39
[processing RopQueryColumnsAll](#) 42

- [processing RopQueryPosition](#) 40
- [processing RopQueryRows](#) 38
- [processing RopResetTable](#) 44
- [processing RopRestrict](#) 37
- [processing RopSeekRow](#) 40
- [processing RopSeekRowBookmark](#) 40
- [processing RopSeekRowFractional](#) 41
- [processing RopSetCollapseState](#) 45
- [processing RopSetColumns](#) 36
- [processing RopSortTable](#) 37
- Setting the columns on a table example
 - [client request buffer](#) 49
 - [server response to client request](#) 50
- Sorting a table by time delivered example
 - [client request buffer](#) 50
 - [server response to client request](#) 51
- [Standards assignments](#) 11

T

- [Table notifications](#) 10
- Table ROP constants
 - [asynchronous flags](#) 14
 - [custom bookmarks](#) 14
 - [predefined bookmarks](#) 13
 - [TableStatus](#) 14
- Table ROPs
 - [RopAbort semantics](#) 19
 - [RopCollapseRow semantics](#) 28
 - [RopCreateBookmark semantics](#) 24
 - [RopExpandRow semantics](#) 27
 - [RopFindRow semantics](#) 24
 - [RopFreeBookmark semantics](#) 26
 - [RopGetCollapseState semantics](#) 28
 - [RopGetStatus semantics](#) 19
 - [RopQueryColumnsAll semantics](#) 24
 - [RopQueryPosition semantics](#) 20
 - [RopQueryRows semantics](#) 18
 - [RopResetTable semantics](#) 27
 - [RopRestrict semantics](#) 17
 - [RopSeekRow semantics](#) 20
 - [RopSeekRowBookmark semantics](#) 21
 - [RopSeekRowFractional semantics](#) 23
 - [RopSetCollapseState semantics](#) 29
 - [RopSetColumns semantics](#) 14
 - [RopSortTable semantics](#) 15
 - [table ROP constants - asynchronous flags](#) 14
 - [table ROP constants - custom bookmarks](#) 14
 - [table ROP constants - predefined bookmarks](#) 13
 - [table ROP constants - TableStatus](#) 14
- [Table ROPs message](#) 13
- Table-specific properties
 - [PidTagContentCount](#) 13
 - [PidTagContentUnreadCount](#) 13
 - [PidTagDepth](#) 13
 - [PidTagInstanceNum](#) 12
 - [PidTagInstID](#) 12
 - [PidTagRowType](#) 12
- [Table-Specific Properties message](#) 12
- [TableStatus](#) 14
- Timer events
 - [client](#) 35

- [server](#) 46
- Timers
 - [client](#) 31
 - [server](#) 35
- [Tracking changes](#) 63
- [Transport](#) 12
- Triggered events - client
 - [advancing the table](#) 33
 - [getting table state](#) 34
 - [preparing the table](#) 31
 - [querying the table](#) 33
 - [registering for notifications](#) 34
- Triggered events - higher-layer
 - [server](#) 35

V

- [Vendor-extensible fields](#) 11
- [Versioning](#) 11

W

- Working with categories example
 - [expanding a category row - client request buffer](#) 55
 - [expanding a category row - overview](#) 55
 - [expanding a category row - server response to client request](#) 56
 - [querying rows with category view - client request buffer](#) 56
 - [querying rows with category view - overview](#) 56
 - [querying rows with category view - server response to client request](#) 57
 - [sorting a table by category - client request buffer](#) 53
 - [sorting a table by category - overview](#) 53
 - [sorting a table by category - server response to client request](#) 54