[MS-OXCPERM]:
Exchange Access and Operation Permissions Protocol Specification

---

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Updated references.
12/03/2008	1.03		Minor editorial fixes.
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	3.0.1	Editorial	Revised and edited the technical content.
02/10/2010	3.1.0	Minor	Updated the technical content.
05/05/2010	3.2.0	Minor	Updated the technical content.
08/04/2010	3.2.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/03/2010	3.2.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the Exchange Access and Operation Permissions protocol, which is used by clients to retrieve and set **permissions** on a **Folder object**.

1.1 Glossary

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

appointment
binary large object (BLOB)
Calendar folder
EntryID
folder
Folder object
handle
little-endian
LogonID
message
Message object
permissions
property (1)
remote operation (ROP)
ROP request
ROP request buffer
ROP response
ROP response buffer
remote procedure call (RPC)
Table object
Server object

The following terms are specific to this document:

Anonymous Client: A client that has connected to the server without providing any user credentials

Default User: A client that has connected with the credentials of a user who does not have an entry in the **Permissions List**.

Permissions List: A list of users and the **permissions** for each of those users.

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-NSPI] Microsoft Corporation, "Name Service Provider Interface (NSPI) Protocol Specification", April 2008, http://msdn.microsoft.com/en-us/library/dd942204(PROT.10).aspx

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

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

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

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

[MS-OXCTABL] Microsoft Corporation, "Table Object Protocol Specification", April 2008.

[MS-OXOSFLD] Microsoft Corporation, "Special Folders Protocol Specification", April 2008.

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

[MS-OXWAVLS] Microsoft Corporation, "Availability Web Service Protocol Specification", April 2008.

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

1.2.2 Informative References

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

1.3 Overview

The Exchange Access and Operation Permissions protocol is used by a client to retrieve and to set a **Permissions List** on a **folder**, as specified in [MS-OXCFOLD], that is stored by the server.

1.3.1 Permissions Table

Figure 1 shows the **message** sequence that is used to retrieve the current Permissions List.

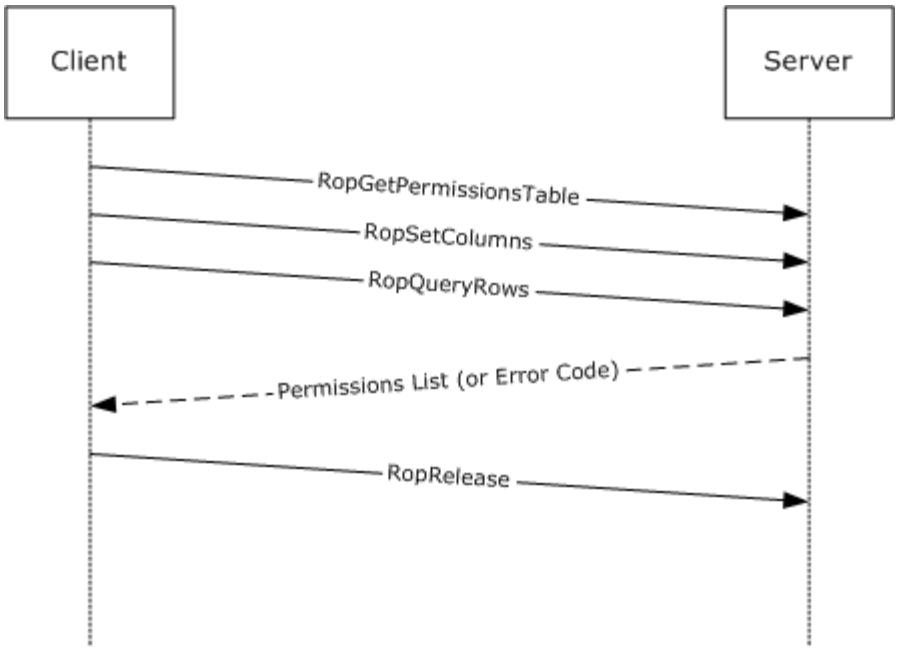


Figure 1: Sequence for retrieving folder permissions

The client sends the **handle** to the Folder object in the RopGetPermissionsTable message that is defined in section 2.2.1.1. This message can be batched together with the RopSetColumns and RopQueryRows **ROP requests** as specified in [MS-OXCTABL]. The server returns a handle to the **Table object**, along with a set of **table** rows that contain the Permissions List for the folder. For a list of common error return values, see [MS-OXPROPS]. After the client is finished reading the Permissions List, it sends a RopRelease message to the server to release the Table object handle.

The message sequence that is used to set the access permissions is shown in Figure 2.

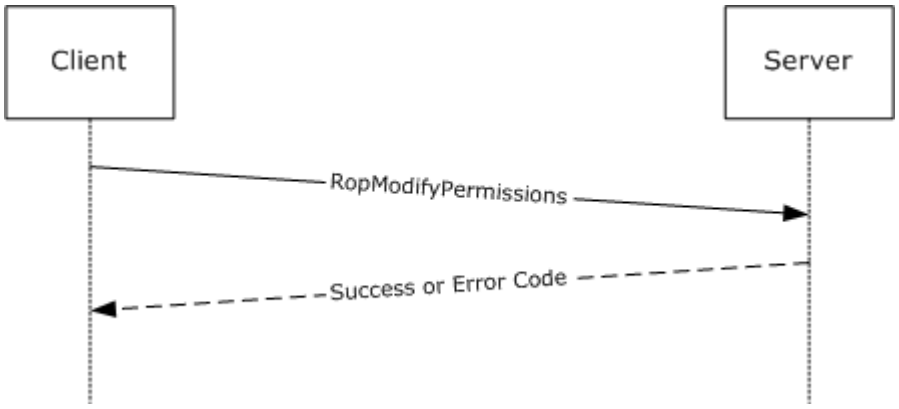


Figure 2: Sequence for setting folder permissions

The client builds a set of table rows that contain modifications to the Permissions List and sends them along with the handle to the Folder object to the server in the RopModifyPermissions message that is defined in section 2.2.1.2. No Table object handle is created by the server when it modifies the permissions, so the client does not send a RopRelease message in this scenario. If the user does not have FolderOwner permissions, the server will return an error result.

1.4 Relationship to Other Protocols

This protocol extends the Folder Object protocol [MS-OXCFOLD] by adding the ability to manage the Permissions List on the folder. If the client and the server both implement the Availability Web Service protocol [MS-OXWAVLS], this protocol also extends that protocol.

This protocol depends on the Remote Operations (ROP) List and Encoding Structure protocol [MS-OXCROPS], the Table Object protocol [MS-OXCTABL], and the Data Structures protocol [MS-OXCDATA] to construct the ROP requests and interpret the **ROP responses**.

1.5 Prerequisites/Preconditions

In addition to the prerequisites of the Folder Object protocol [MS-OXCFOLD], the Exchange Access and Operation Permissions protocol requires that the client be connected to the server by using credentials that belong to a user that has FolderVisible permissions for the folder to read the Permissions List, and FolderOwner permissions to modify the Permissions List.

Before sending any of these requests to the server, the client MUST have successfully logged on to the server by using RopLogon, as specified in [MS-OXCROPS] section 2.2.3.1, and have a valid **LogonID**.

The client MUST have sent a RopOpenFolder request and received a handle to the Folder object on the server. This handle will be included in the request buffers for the ROP requests that are used in this protocol.

1.6 Applicability Statement

A client can use the Exchange Access and Operation Permissions protocol any time it needs to read or write the Permissions List on a folder. For example, the client might enable another user to view a folder by adding an entry that has read permissions for that user to the Permissions List on the folder.

1.7 Versioning and Capability Negotiation

This protocol does an explicit capability negotiation as specified in this section.

This protocol can be used to extend the Availability Web Service protocol [MS-OXWAVLS] when retrieving or setting permissions on the **Calendar folder** as specified in [MS-OXOSFLD] by retrieving and setting additional permissions that affect the behavior of the Availability Web Service protocol. The client first checks the version number returned by the server in the results from **EcDoConnectEx**, as specified in [MS-OXCRPC]. If the server returns a version that is greater than or equal to 8.0.360.0,<1> the client includes the **IncludeFreeBusy** flag in the **ROP request buffer** for both the RopGetPermissionsTable and RopModifyPermissions messages. If the server returns a version that is less than 8.0.360.0, the client MUST NOT include the **IncludeFreeBusy** flag in the ROP request buffer. The presence of the **IncludeFreeBusy** flag in the ROP request buffer indicates to the server that the client is capable of extending the Availability Web Service protocol with the **FreeBusySimple** and **FreeBusyDetailed** permissions.

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 the server respectively by using the underlying protocol specified by [MS-OXCROPS].

2.2 Message Syntax

Unless otherwise noted, sizes in this section are expressed in BYTES.

Unless otherwise noted, the fields specified in this section are packed in buffers in the order that they appear in this document, without any padding.

Unless otherwise noted, the fields specified in this section, which are larger than a single BYTE, MUST be converted to **little-endian** order when packed in buffers and converted from little-endian order when unpacked.

2.2.1 Permissions Table

The client MUST send the RopGetPermissionsTable and RopModifyPermissions messages to retrieve and set the Permissions List on a folder.

2.2.1.1 RopGetPermissionsTable

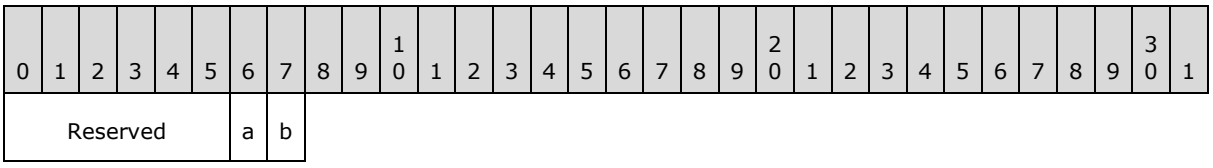
The client sends the RopGetPermissionsTable request to retrieve a **Server object** handle to a Table object. The client uses the as specified in [MS-OXCROPS] as specified in [MS-OXCTABL] to retrieve the current permissions on a folder.

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

2.2.1.1.1 Request Buffer

2.2.1.1.1.1 TableFlags

This is an 8-bit flag structure specified in [MS-OXCROPS]. The flags within this structure are specified in the following table.



Reserved (6 bits): These bits (bitmask 0xFC) are reserved. They MUST be set to 0 by the client and ignored by the server.

a (1 bit): This bit (bitmask 0x02) is the **IncludeFreeBusy** flag. If this bit is set, the server MUST include the values of the **FreeBusySimple** and **FreeBusyDetailed** bits in the PidTagMemberRights property. If this bit is not set, the client MUST ignore the values of those bits. The client SHOULD set this bit if the folder is the Calendar folder as specified in [MS-

OXOSFLD] and the server version is greater than or equal to 8.0.360.0, as specified in [MS-OXCRPC]. The client MUST NOT set this bit in any other circumstances.

b (1 bit): This bit (bitmask 0x01) is reserved. It MUST be set to 0 by the client and ignored by the server.

2.2.1.1.2 Response Buffer

2.2.1.1.2.1 ReturnValue

The **ReturnValue** is a **PtypErrorCode** value that indicates the result of the operation. To indicate success, the server MUST return 0x00000000. For a list of common error return values, see [MS-OXPROPS].

2.2.1.2 RopModifyPermissions

The client sends the RopModifyPermissions request to create, modify, or delete permissions on a folder.

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

2.2.1.2.1 Request Buffer

This section documents the fields in the ROP request buffer that are not fully documented in [MS-OXCROPS]. Other fields that are fully documented in [MS-OXCROPS] include:

- **ModifyCount**,
- **PermissionData**, and within **PermissionData**:
  - **PropertyValueCount**
  - **PropertyValues**

2.2.1.2.1.1 ModifyFlags

This is an 8-bit flag structure specified in [MS-OXCROPS]. The flags within this structure are specified in the following table:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Reserved						a	b																								

Reserved (6 bits): These bits MUST be set to 0 by the client and ignored by the server.

a (1 bit): This bit (bitmask 0x02) is the **IncludeFreeBusy** flag. If this bit is set, the server MUST use the values of the **FreeBusySimple** and **FreeBusyDetailed** bits in the PidTagMemberRights property value when modifying the folder permissions. If this bit is not set, the server MUST ignore the values of those bits. The client SHOULD set this bit if the folder is the Calendar folder as specified in [MS-OXOSFLD] and the server version is greater than or equal to 8.0.360.0 as specified in [MS-OXCRPC]. The client MUST NOT set this bit in any other circumstances.

b (1 bit): This bit (bitmask 0x01) is the **ReplaceRows** flag. If this bit is set, the server MUST replace any existing folder permissions, and the client MUST NOT include any **PermissionsDataFlags** field values other than **AddRow** in this request. If this bit is not set, the server MUST modify the existing folder permissions with the changes in this request (delete, modify, or add). The client SHOULD NOT set this bit.<2>

2.2.1.2.1.2 PermissionDataFlags

The following table specifies the allowed values for the **PermissionDataFlags** field of the **PermissionData** structure, as specified in [MS-OXCROPS].

Name	Value	Description
AddRow	0x01	Adds new permissions that are specified in the **PermissionData** structure.
ModifyRow	0x02	Modifies the existing permissions for a user identified by the value of the PidTagMemberId property.
RemoveRow	0x04	Removes the existing permissions for a user identified by the value of the PidTagMemberId property.

2.2.1.2.1.3 PropertyValues

This section specifies the set of **TaggedPropertyValue** structures as specified in [MS-OXCDATA] that can be included in the **PropertyValues** field of the **PermissionData** structure, as specified in [MS-OXCROPS].

When deleting the entry for a user from the Permissions List, the client MUST include PidTagMemberId. The client MUST NOT include any other property.

When adding an entry for a user to the Permissions List, the client MUST NOT include PidTagMemberId. The client MUST include PidTagEntryId and PidTagMemberRights.

When modifying the permissions for a user, the client MUST NOT include PidTagEntryId. The client MUST include PidTagMemberId and PidTagMemberRights.

For more details about properties, property types, and the buffer format of the **PropertyValue** structure, see [MS-OXCROPS] and [MS-OXCDATA].

2.2.1.2.2 Response Buffer

2.2.1.2.2.1 ReturnValue

The **ReturnValue** is a **PtypErrorCode** value that indicates the result of the operation. To indicate success, the server MUST return 0x00000000. For a list of common error return values, see [MS-OXPROPS].

2.2.1.3 PidTagEntryId

This is a variable-length **BLOB** that contains an **EntryID** for the user in the Permissions List, as specified in [MS-NSPI].

When searching for an existing entry for the user in the Permissions List, the client MUST include this property in the ROP request buffer for the RopSetColumns request before reading rows from the Table object that is returned in the response to RopGetPermissionsTable, and the client MUST use

this property to identify the entry. If the client does not need to match entries in the Permissions List to specific users, the client SHOULD omit this property from the ROP request buffer for the RopSetColumns request.

There is one reserved value for PidTagEntryId, which is the empty BLOB with a length of zero bytes. This value MUST be used with one of the reserved values for PidTagMemberId specified in the following section 2.2.1.4.

2.2.1.4 PidTagMemberId

This is a **PtypInteger64** value that contains a unique identifier that the messaging server generates for each user. The client MUST NOT specify this property when adding permissions for a new user, but MUST specify it when modifying or deleting permissions for a user that already exists in the Permissions List.

The client MUST include this property in the ROP request buffer for the RopSetColumns message before reading rows from the Table object that is returned in the response to RopGetPermissionsTable. The server MUST include this property in those rows.

The following table lists the two reserved values for PidTagMemberId.

Value	Description
0xFFFFFFFFFFFFFFFF	**Anonymous Client**: The server MUST use the permissions specified in PidTagMemberRights for any anonymous users that have not been authenticated with user credentials.
0x0000000000000000	**Default User**: The server MUST use the permissions specified in PidTagMemberRights for any users that have been authenticated with user credentials but do not appear anywhere else in the Permissions List.

2.2.1.5 PidTagMemberName

This is a string property that is the user-readable name of the user.

When displaying the contents of the Permissions List, the client SHOULD include this property in the ROP request buffer for the RopSetColumns request before reading rows from the Table object returned in the response to RopGetPermissionsTable. For any entries in the Permissions List that are not reserved (as specified in section 2.2.1.4), the server MUST provide a value for this string property that the client can use to display the entry to the user.<3> If the client is not displaying the contents of the Permissions List, it SHOULD omit this property from the ROP request buffer for the RopSetColumns request.

2.2.1.6 PidTagMemberRights

This is a **PtypInteger32** flag structure that contains the permissions for the specified user. When reading the Permissions List with a RopGetPermissionsTable request, this property reflects the permissions that have actually been set on the folder.

The flags within this structure are defined in the following table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved																			a	b	c	d	e	f	g	h	i	j	k	l	m

Reserved (19 bits): These bits (bitmask 0xFFFFE000) are reserved. The client and server MUST NOT set these flags.

a (1 bit): This bit (bitmask 0x00001000) is the **FreeBusyDetailed** flag. If the **IncludeFreeBusy** flag was included in the **ModifyFlags** or **TableFlags** field of the ROP request buffer, this flag indicates that the server MUST allow the specified user's client to retrieve detailed information about the **appointments** on the calendar through the Availability Web Service protocol, as specified in [MS-OXWAVLS]. Otherwise, the server MUST NOT allow the specified user's client to see these details. If the client sets this flag, it MUST also set the **FreeBusySimple** flag.

b (1 bit): This bit (bitmask 0x00000800) is the **FreeBusySimple** flag. If the **IncludeFreeBusy** flag was included in the ROP request buffer, this flag indicates that the server MUST allow the specified user's client to retrieve information through the Availability Web Service protocol, as specified in [MS-OXWAVLS]. Otherwise, the server MUST NOT allow the specified user's client to retrieve information through the Availability Web Service protocol.<4>

c (1 bit): This bit (bitmask 0x00000400) is the **FolderVisible** flag. If this flag is set, it indicates that the server MUST allow the specified user's client to see the folder in the folder hierarchy table and request a handle for the folder by using a RopOpenFolder request, as specified in [MS-OXCFOLD]. If the client sets the **ReadAny** flag or the **FolderOwner** flag, the server MUST set this flag as well.

d (1 bit): This bit (bitmask 0x00000200) is the **FolderContact** flag. If this flag is set, it indicates that the server MUST include the specified user in any list of administrative contacts associated with the folder. If this flag is not set, the server MUST NOT include the specified user in any such list. If neither this flag nor the **FolderOwner** flag is set, the specified user's client SHOULD NOT display the Permissions List for this folder.<5> If the client sets this flag for the reserved Default User or Anonymous Client entries, the server MUST ignore this flag and it MUST NOT apply this flag to those entries.

e (1 bit): This bit (bitmask 0x00000100) is the **FolderOwner** flag. If this flag is set, the server MUST allow the specified user's client to modify properties set on the folder itself, including the folder permissions. If this flag is not set, the server MUST NOT allow the specified user's client to make those modifications.

f (1 bit): This bit (bitmask 0x00000080) is the **CreateSubFolder** flag. If this flag is set, the server MUST allow the specified user's client to create new folders within the folder. If this flag is not set, the server MUST NOT allow the user's client to create new folders within the folder.

g (1 bit): This bit (bitmask 0x00000040) is the **DeleteAny** flag. If this flag is set, the server MUST allow the specified user's client to delete any **Message object** in the folder. If this flag is not set, the server MUST NOT allow the user's client to delete Message objects that are owned by other users.

h (1 bit): This bit (bitmask 0x00000020) is the **EditAny** flag. If this flag is set, the server MUST allow the specified user's client to modify any Message object in the folder. If this flag is not set, the server MUST NOT allow the user's client to modify Message objects that are owned by other users.

i (1 bit): This bit (bitmask 0x00000010) is the **DeleteOwned** flag. If this flag is set, the server MUST allow the specified user's client to delete any Message object in the folder that was created by that user. If this flag is not set, the server MUST NOT allow the user's client to delete Message objects that were created by that user. If the client sets the DeleteAny flag, the server MUST set this flag as well.

j (1 bit): This bit (bitmask 0x00000008) is the **EditOwned** flag. If this flag is set, the server MUST allow the specified user's client to modify any Message object in the folder that was created by that user. If this flag is not set, the server MUST NOT allow the user's client to modify Message objects that were created by that user. If the client sets the **EditAny** flag, the server MUST set this flag as well.

k (1 bit): This bit (bitmask 0x00000004) is reserved. The client and server MUST NOT set this flag, and the server MUST ignore it if it is set.

l (1 bit): This bit (bitmask 0x00000002) is the **Create** flag. If this flag is set, the server MUST allow the specified user's client to create new Message objects in the folder. If this flag is not set, the server MUST NOT allow the user's client to create new Message objects in the folder.

m (1 bit): This bit (bitmask 0x00000001) is the **ReadAny** flag. If this flag is set, the server MUST allow the specified user's client to read any Message object in the folder. If this flag is not set, the server MUST NOT allow the user's client to read Message objects that are owned by other users.The client MUST include this property in the ROP request buffer for the RopSetColumns message before reading rows from the Table object returned in the response to RopGetPermissionsTable. The server MUST include this property in those rows.

3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

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

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Retrieving Folder Permissions

Before retrieving the Permissions List from a folder, the client can try to open a stream handle on the PidTagSecurityDescriptorAsXml property of the folder by using a RopOpenStream request.<6> If the client sends this request, the server MUST return an error code of **ecNotImplemented** instead of satisfying the request.

To retrieve the current Permissions List from a folder, the client SHOULD send the following **ROP** requests to the server:

- RopGetPermissionsTable, as specified in section 2.2.1.1.
- RopSetColumns, with a column set that includes some or all of the following properties:
  - PidTagEntryId, as specified in section 2.2.1.3.
  - PidTagMemberId, as specified in section 2.2.1.4.
  - PidTagMemberName, as specified in section 2.2.1.5.
  - PidTagMemberRights, as specified in section 2.2.1.6.
- RopQueryRows, as specified in [MS-OXCTABL].

If all the ROP requests succeed, the result buffer for the RopQueryRows ROP MUST contain a PropertyRowSet structure, as specified in [MS-OXCDATA]. The client MUST extract the Permissions List from the properties in the **PropertyRowSet**.

If the RopGetPermissionsTable request succeeded, the client MUST release the Table object by sending a RopRelease request to the server.

3.1.4.2 Adding Folder Permissions

To add new permissions to the Permissions List, the client MUST send a RopModifyPermissions request to the server. The client MUST use **AddRow** for the **PermissionDataFlags** field on any **PermissionData** structure in the ROP request buffer. The client MUST include **PropertyValue** structures in the **PermissionData** structure that contains values for the following **properties**:

- PidTagEntryId
- PidTagMemberRights

The client MAY include the **ReplaceRows** flag in the RopModifyPermissions ROP request buffer,<7> in which case all of the **PermissionData** structures in the ROP request buffer MUST use **AddRow** for the **PermissionDataFlags** field. If the client does not include the **ReplaceRows** flag, it MUST retrieve the existing Permissions List as specified in section 3.1.4.1 and use the PidTagMemberId property to modify the permissions for any users that already exist in the Permissions List, as specified in section 3.1.4.3.

3.1.4.3 Modifying Folder Permissions

To modify existing permissions in the Permissions List, the client MUST retrieve the existing Permissions List as specified in section 3.1.4.1 to get the value of the PidTagMemberId property that is assigned to the specified user in the Permissions List.

The client MUST send a RopModifyPermissions request to the server. The client MUST use **ModifyRow** for the **PermissionDataFlags** field on any **PermissionData** structure in the ROP request buffer. The client MUST include **PropertyValue** structures in the **PermissionData** structure that contains values for the following properties:

- PidTagMemberId
- PidTagMemberRights

3.1.4.4 Removing Folder Permissions

To remove permissions in the Permissions List, the client MUST retrieve the existing Permissions List as specified in section 3.1.4.1 to get the value of the PidTagMemberId property that is assigned to the specified user in the Permissions List.

The client MUST send a RopModifyPermissions request to the server. The client MUST use **RemoveRow** for the **PermissionDataFlags** field on any **PermissionData** structure in the ROP request buffer. The client MUST include **PropertyValue** structures in the **PermissionData** structure containing values for the PidTagMemberId property.

3.1.5 Message Processing Events and Sequencing Permissions

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

The Abstract Data Model for the client and server roles is the same.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Accessing Folders

When a client sends a request to the server to access a folder, as specified in [MS-OXCFOLD], the server MUST allow or deny the request based on the Permissions List for the folder and any user credentials that the client provided when making the request. If the client did not provide any user credentials, the server MUST use the permissions that have been set for the Anonymous Client in the Permissions List. If the client provided user credentials for a user that is included in the Permissions List, either explicitly or through membership in a group that is included in the Permissions List, the server MUST use the permissions for that user. If the client provided user credentials for a user that is not otherwise included in the Permissions List, the server MUST use the permissions for the Default User.

3.2.5 Message Processing Events and Sequencing Permissions

3.2.5.1 RopGetPermissionsTable

When the server receives a RopGetPermissionsTable request from the client, the server MUST check to see if the user has been assigned **FolderVisible** permissions for the folder. If the user does not have **FolderVisible** permissions, the server MUST return an error result. If the user does have **FolderVisible** permissions, the server MUST return a Server object handle to a Table object that can be used to retrieve the Permissions List from the folder, as defined in section 3.1.4.1.

3.2.5.2 RopModifyPermissions

When the server receives a RopModifyPermissions request from the client, the server MUST check to see if the user has been assigned **FolderOwner** permissions for the folder. If the user does not have **FolderOwner** permissions, the server MUST return an error result. If the user does have **FolderOwner** permissions, the server MUST update the Permissions List for the folder with the **PermissionData** structures in the ROP request buffer, as specified in section 2.2.1.2.1.

3.2.5.3 Reading PidTagSecurityDescriptorAsXml

When the server receives a RopOpenStream request for the PidTagSecurityDescriptorAsXml string property on a folder, the server MUST return an error code of **ecNotImplemented** rather than satisfying the request, as defined in section 3.1.4.1.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Adding an Entry for "User8" to the Permissions List

In this example, the client is adding an entry for "user8" to the Permissions List on the Calendar folder. The client in this example starts by trying to read the deprecated PidTagSecurityDescriptorAsXml string property, as defined in section 3.2.5.3. The client sends the following request:

RopOpenStream

RopId: 0x2B

LogonID: 0

InputHandleIndex: 1 (HSOT=0x000001DA)

OutputHandleIndex: 2 (HSOT=0xFFFFFFFF)

PropertyTag:

PidTagSecurityDescriptorAsXml:0x0E6A001F (PT_UNICODE)

OpenModeFlags: 0x00 ReadOnly rights

```
9 bytes total in this ROP
rop(1), iLogon(1), iHsot(1), data
0000: 2B 00 01 02 1F 00 6A 0E-00
```

The server returns the following response buffer, which indicates that it does not support the PidTagSecurityDescriptorAsXml string property on this folder.

RopOpenStream

RopId: 0x2B

InputHandleIndex: 2 (HSOT=0xFFFFFFFF)

ReturnValue: ecNotImplemented (0x80040102)

```
6 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 2B 02 02 01 04 80
```

The client in this example falls back to using RopGetPermissionsTable as defined in section 3.1.4.1. The client sends the following ROP requests, batched together into a single **RPC**.

RopGetPermissionsTable

RopId: 0x3E

LogonID: 0

InputHandleIndex: 0 (HSOT=0x000001DA)

NewPermissionsHandleIndex: 1 (HSOT=0xFFFFFFFF)

TableFlags: 0x02 IncludeFreeBusy

```
5 bytes total in this ROP
rop(1), iLogon(1), iHsot(1), data
0000: 3E 00 00 01 02
```

RopSetColumns

RopId: 0x12

LogonID: 0

InputHandleIndex: 1 (HSOT=0xFFFFFFFF)

WantAsync: 0x00 Wait

PropertyValueCount: 4 (0x04)

PidTagMemberId:0x66710014 (PT_I8)

PidTagMemberName:0x6672001F (PT_UNICODE)

PidTagMemberRights:0x66730003 (PT_LONG)

PidTagEntryId:0x0FFF0102 (PT_BINARY)

```
22 bytes total in this ROP
rop(1), iLogon(1), iHsot(1), data
0000: 12 00 01 00 04 00 14 00-71 66 1F 00 72 66 03 00 ........qf..rf..
0010: 73 66 02 01 FF 0F                              sf....
```

RopQueryRows

RopId: 0x15

LogonID: 0

InputHandleIndex: 1 (HSOT=0xFFFFFFFF)

WantCurrentRow: 0 - Advance

WantForwardRead: 1 - True

RowCount: 0x1000 (4096)

```
7 bytes total in this ROP
rop(1), iLogon(1), iHsot(1), data
0000: 15 00 01 00 01 00 10
```

The server returns the following response buffer with the PermissionData structures that contain the current Permissions List:

RopGetPermissionsTable

RopId: 0x3E

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

```
6 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 3E 01 00 00 00 00
```

RopSetColumns

RopId: 0x12

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

CompletionStatus: TBLSTAT_COMPLETE (0x00)

```
7 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 12 01 00 00 00 00 00
```

RopQueryRows

RopId: 0x15

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

bookmark: 0x02 BOOKMARK_END

RowCount: 2

RowPropertyArray:

RowPropertyArray[1/2]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId:0x66710014 (PT_I8)

0x0000000000000000

PidTagMemberName:0x6672001F (PT_UNICODE)

"" (null)

PidTagMemberRights:0x66730003 (PT_LONG)

0x00000800 (2048)

PidTagEntryId:0x0FFF0102 (PT_BINARY)

0 Bytes

RowPropertyArray[2/2]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId:0x66710014 (PT_I8)

0xFFFFFFFFFFFFFFFF

PidTagMemberName:0x6672001F (PT_UNICODE)

"Anonymous"

PidTagMemberRights:0x66730003 (PT_LONG)

0x00000000 (0)

PidTagEntryId:0x0FFF0102 (PT_BINARY)

0 Bytes

```
61 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 15 01 00 00 00 00 02 02-00 00 00 00 00 00 00 00 ................
0010: 00 00 00 00 00 08 00 00-00 00 00 FF FF FF FF FF ................
0020: FF FF FF 41 00 6E 00 6F-00 6E 00 79 00 6D 00 6F ...A.n.o.n.y.m.o
0030: 00 75 00 73 00 00 00 00-00 00 00 00 00          .u.s.........
```

The Permissions List on this folder starts out with two entries. The Default User entry has the **FreeBusySimple** permissions (0x00000800) in this folder. The Anonymous Client entry has no permissions (0x00000000) in this folder.

The client sends the following RopModifyPermissions request to add "user8" to the Permissions List with **FreeBusyDetailed**, **FreeBusySimple**, **FolderVisible**, **FolderContact**, **FolderOwner**, **CreateSubFolder**, **DeleteAny**, **EditAny**, **DeleteOwned**, **EditOwned**, **Create**, and **ReadAny** permissions (0x00001FFB) in this folder:

RopModifyPermissions

RopId: 0x40

LogonID: 0

InputHandleIndex: 2 (HSOT=0x000001da)

ModifyFlags: 0x02 IncludeFreeBusy

ModifyCount: 1

Parsing row: 1

PermissionsDataFlags: 0x01 AddRow

PropertyValueCount: 2 (0x02)

PidTagEntryId:0x0FFF0102 (PT_BINARY)

```
124 Bytes
0000: 00 00 00 00 DC A7 40 C8-C0 42 10 1A B4 B9 08 00 ......@..B......
0010: 2B 2F E1 82 01 00 00 00-00 00 00 00 2F 6F 3D 46 +/............/o=F
0020: 69 72 73 74 20 4F 72 67-61 6E 69 7A 61 74 69 6F irst Organizatio
0030: 6E 2F 6F 75 3D 45 78 63-68 61 6E 67 65 20 41 64 n/ou=Exchange Ad
0040: 6D 69 6E 69 73 74 72 61-74 69 76 65 20 47 72 6F ministrative Gro
0050: 75 70 20 28 46 59 44 49-42 4F 48 46 32 33 53 50 up (FYDIBOHF23SP
0060: 44 4C 54 29 2F 63 6E 3D-52 65 63 69 70 69 65 6E DLT)/cn=Recipien
0070: 74 73 2F 63 6E 3D 75 73-65 72 38 00             ts/cn=user8.
```

PidTagMemberRights:0x66730003 (PT_LONG)

0x00001FFB (8187)

```
147 bytes total in this ROP
rop(1), iLogon(1), iHsot(1), data
0000: 40 00 02 02 01 00 01 02-00 02 01 FF 0F 7C 00 00 @............|..
0010: 00 00 00 DC A7 40 C8 C0-42 10 1A B4 B9 08 00 2B .....@..B......+
0020: 2F E1 82 01 00 00 00 00-00 00 00 2F 6F 3D 46 69 /........../o=Fi
0030: 72 73 74 20 4F 72 67 61-6E 69 7A 61 74 69 6F 6E rst Organization
0040: 2F 6F 75 3D 45 78 63 68-61 6E 67 65 20 41 64 6D /ou=Exchange Adm
0050: 69 6E 69 73 74 72 61 74-69 76 65 20 47 72 6F 75 inistrative Grou
0060: 70 20 28 46 59 44 49 42-4F 48 46 32 33 53 50 44 p (FYDIBOHF23SPD
0070: 4C 54 29 2F 63 6E 3D 52-65 63 69 70 69 65 6E 74 LT)/cn=Recipient
0080: 73 2F 63 6E 3D 75 73 65-72 38 00 03 00 73 66 FB s/cn=user8...sf.
0090: 1F 00 00                                       ...
```

The server returns the following response buffer, which indicates that it has successfully updated the Permissions List for the folder:

RopModifyPermissions

RopId: 0x40

InputHandleIndex: 2 (HSOT=0x000001DA)

ReturnValue: ecNone (success) (0x00000000)

```
6 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 40 02 00 00 00 00
```

4.2 Modifying the Entry for "User8 " in the Permissions List

In this example, the client is modifying the entry for "user8" in the Permissions List on the Calendar folder. First the client needs to read the Permissions List to get the value of the PidTagMemberId property for "user8", as defined in section 3.1.4.1. The client sends the same RopGetPermissionsTable, RopSetColumns, and RopQueryRows requests as in the example in section 4.1. The server returns the following response buffer with the **PermissionData** structures that contain the current Permissions List:

RopGetPermissionsTable

RopId: 0x3E

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

```
6 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 3E 01 00 00 00 00
```

RopSetColumns

RopId: 0x12

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

CompletionStatus: TBLSTAT_COMPLETE (0x00)

```
7 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 12 01 00 00 00 00 00
```

RopQueryRows

RopId: 0x15

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

Bookmark: 0x02 BOOKMARK_END

RowCount: 3

RowPropertyArray:

RowPropertyArray[1/3]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId: 0x66710014 (PT_I8)

0x0000000000000000

PidTagMemberName: 0x6672001F (PT_UNICODE)

"" (null)

PidTagMemberRights: 0x66730003 (PT_LONG)

0x00000800 (2048)

PidTagEntryId: 0x0FFF0102 (PT_BINARY)

0 Bytes

RowPropertyArray[2/3]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId: 0x66710014 (PT_I8)

0x0000001500000002

PidTagMemberName: 0x6672001F (PT_UNICODE)

"user8"

PidTagMemberRights: 0x66730003 (PT_LONG)

0x00001FFB (8187)

PidTagEntryId: 0x0FFF0102 (PT_BINARY)

```
124 Bytes
0000: 00 00 00 00 DC A7 40 C8-C0 42 10 1A B4 B9 08 00 ......@..B......
0010: 2B 2F E1 82 01 00 00 00-00 00 00 00 2F 4F 3D 46 +/........../O=F
0020: 49 52 53 54 20 4F 52 47-41 4E 49 5A 41 54 49 4F IRST ORGANIZATIO
0030: 4E 2F 4F 55 3D 45 58 43-48 41 4E 47 45 20 41 44 N/OU=EXCHANGE AD
0040: 4D 49 4E 49 53 54 52 41-54 49 56 45 20 47 52 4F MINISTRATIVE GRO
0050: 55 50 20 28 46 59 44 49-42 4F 48 46 32 33 53 50 UP (FYDIBOHF23SP
0060: 44 4C 54 29 2F 43 4E 3D-52 45 43 49 50 49 45 4E DLT)/CN=RECIPIEN
0070: 54 53 2F 43 4E 3D 55 53-45 52 38 00             TS/CN=USER8.
```

RowPropertyArray[3/3]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId: 0x66710014 (PT_I8)

0xFFFFFFFFFFFFFFFF

PidTagMemberName: 0x6672001F (PT_UNICODE)

"Anonymous"

PidTagMemberRights: 0x66730003 (PT_LONG)

0x00000000 (0)

PidTagEntryId: 0x0FFF0102 (PT_BINARY)

0 Bytes

```
212 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 15 01 00 00 00 00 02 03-00 00 00 00 00 00 00 00 ................
0010: 00 00 00 00 00 08 00 00-00 00 00 02 00 00 00 15 ................
0020: 00 00 00 75 00 73 00 65-00 72 00 38 00 00 00 FB ...u.s.e.r.8....
0030: 1F 00 00 7C 00 00 00 00-00 DC A7 40 C8 C0 42 10 ...|.......@..B.
0040: 1A B4 B9 08 00 2B 2F E1-82 01 00 00 00 00 00 00 .....+/.........
0050: 00 2F 4F 3D 46 49 52 53-54 20 4F 52 47 41 4E 49 ./O=FIRST ORGANI
0060: 5A 41 54 49 4F 4E 2F 4F-55 3D 45 58 43 48 41 4E ZATION/OU=EXCHAN
0070: 47 45 20 41 44 4D 49 4E-49 53 54 52 41 54 49 56 GE ADMINISTRATIV
0080: 45 20 47 52 4F 55 50 20-28 46 59 44 49 42 4F 48 E GROUP (FYDIBOH
0090: 46 32 33 53 50 44 4C 54-29 2F 43 4E 3D 52 45 43 F23SPDLT)/CN=REC
00a0: 49 50 49 45 4E 54 53 2F-43 4E 3D 55 53 45 52 38 IPIENTS/CN=USER8
00b0: 00 00 FF FF FF FF FF FF-FF FF 41 00 6E 00 6F 00 ..........A.n.o.
00c0: 6E 00 79 00 6D 00 6F 00-75 00 73 00 00 00 00 00 n.y.m.o.u.s.....
00d0: 00 00 00 00                                     ....
```

The Permissions List on this folder now has an entry for "user8" with the PidTagMemberRights property value of 0x00001FFB that the client set in the previous example. The value of the PidTagMemberId property for "user8" is 0x0000001500000002. With that property value, the client can now send the following RopModifyPermissions request to change the permissions for "user8" from 0x00001FFB to 0x00001800 (**FreeBusyDetailed** and **FreeBusySimple**):

RopModifyPermissions

RopId: 0x40

LogonID: 0

InputHandleIndex: 0 (HSOT=0x000001DA)

ModifyFlags: 0x02 IncludeFreeBusy

ModifyCount: 1

Parsing row: 1

PermissionsDataFlags: 0x02 ModifyRow

PropertyValueCount: 2 (0x02)

PidTagMemberId: 0x66710014 (PT_I8)

0x0000001500000002

PidTagMemberRights: 0x66730003 (PT_LONG)

0x00001800 (6144)

```
29 bytes total in this ROP
rop(1), iLogon(1), iHsot(1), data
0000: 40 00 00 02 01 00 02 02-00 14 00 71 66 02 00 00 @..........qf...
0010: 00 15 00 00 00 03 00 73-66 00 18 00 00          .......sf....
```

The server returns the following response buffer, which indicates that it successfully updated the Permissions List for the folder :

RopModifyPermissions

RopId: 0x40

InputHandleIndex: 0 (HSOT=0x000001DA)

ReturnValue: ecNone (success) (0x00000000)

```
6 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 40 00 00 00 00 00
```

4.3 Removing the Entry for "User8" in the Permissions List

In this example, the client is removing the entry for "user8 " from the Permissions List on the Calendar folder. First, the client needs to read the Permissions List to get the value of the PidTagMemberId property for "user8", as defined in section 3.1.4.1. The client sends the same RopGetPermissionsTable, RopSetColumns, and RopQueryRows requests as in the example in section 4.1. The server returns the following response buffer with the **PermissionData** structures that contain the current Permissions List:

RopGetPermissionsTable

RopId: 0x3E

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

```
6 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 3E 01 00 00 00 00
```

RopSetColumns

RopId: 0x12

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

CompletionStatus: TBLSTAT_COMPLETE (0x00)

```
7 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 12 01 00 00 00 00 00
```

RopQueryRows

RopId: 0x15

InputHandleIndex: 1 (HSOT=0x000000CA)

ReturnValue: ecNone (success) (0x00000000)

Bookmark: 0x02 BOOKMARK_END

RowCount: 3

RowPropertyArray:

RowPropertyArray[1/3]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId: 0x66710014 (PT_I8)

0x0000000000000000

PidTagMemberName: 0x6672001F (PT_UNICODE)

"" (null)

PidTagMemberRights: 0x66730003 (PT_LONG)

0x00000800 (2048)

PidTagEntryId: 0x0FFF0102 (PT_BINARY)

0 Bytes

RowPropertyArray[2/3]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId: 0x66710014 (PT_I8)

0x0000001500000002

PidTagMemberName: 0x6672001F (PT_UNICODE)

"user8"

PidTagMemberRights: 0x66730003 (PT_LONG)

0x00001800 (6144)

PidTagEntryId: 0x0FFF0102 (PT_BINARY)

```
124 Bytes
0000: 00 00 00 00 DC A7 40 C8-C0 42 10 1A B4 B9 08 00 ......@..B......
0010: 2B 2F E1 82 01 00 00 00-00 00 00 00 2F 4F 3D 46 +/.........../O=F
0020: 49 52 53 54 20 4F 52 47-41 4E 49 5A 41 54 49 4F IRST ORGANIZATIO
0030: 4E 2F 4F 55 3D 45 58 43-48 41 4E 47 45 20 41 44 N/OU=EXCHANGE AD
0040: 4D 49 4E 49 53 54 52 41-54 49 56 45 20 47 52 4F MINISTRATIVE GRO
0050: 55 50 20 28 46 59 44 49-42 4F 48 46 32 33 53 50 UP (FYDIBOHF23SP
0060: 44 4C 54 29 2F 43 4E 3D-52 45 43 49 50 49 45 4E DLT)/CN=RECIPIEN
0070: 54 53 2F 43 4E 3D 55 53-45 52 38 00             TS/CN=USER8.
```

RowPropertyArray[3/3]:

HasError: 0

PropertyArray:

PropertyValueCount: 4

PidTagMemberId: 0x66710014 (PT_I8)

0xFFFFFFFFFFFFFFFF

PidTagMemberName: 0x6672001F (PT_UNICODE)

"Anonymous"

PidTagMemberRights: 0x66730003 (PT_LONG)

0x00000000 (0)

PidTagEntryId: 0x0FFF0102 (PT_BINARY)

0 Bytes

```
212 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 15 01 00 00 00 00 02 03-00 00 00 00 00 00 00 00 ................
0010: 00 00 00 00 00 08 00 00-00 00 00 02 00 00 00 15 ................
0020: 00 00 00 75 00 73 00 65-00 72 00 38 00 00 00 00 ...u.s.e.r.8....
0030: 18 00 00 7C 00 00 00 00-00 DC A7 40 C8 C0 42 10 ...|.......@..B.
0040: 1A B4 B9 08 00 2B 2F E1-82 01 00 00 00 00 00 00 .....+/.........
0050: 00 2F 4F 3D 46 49 52 53-54 20 4F 52 47 41 4E 49 ./O=FIRST ORGANI
0060: 5A 41 54 49 4F 4E 2F 4F-55 3D 45 58 43 48 41 4E ZATION/OU=EXCHAN
0070: 47 45 20 41 44 4D 49 4E-49 53 54 52 41 54 49 56 GE ADMINISTRATIV
0080: 45 20 47 52 4F 55 50 20-28 46 59 44 49 42 4F 48 E GROUP (FYDIBOH
0090: 46 32 33 53 50 44 4C 54-29 2F 43 4E 3D 52 45 43 F23SPDLT)/CN=REC
```

```
00a0: 49 50 49 45 4E 54 53 2F-43 4E 3D 55 53 45 52 38 IPIENTS/CN=USER8
00b0: 00 00 FF FF FF FF FF FF-FF FF 41 00 6E 00 6F 00 ..........A.n.o.
00c0: 6E 00 79 00 6D 00 6F 00-75 00 73 00 00 00 00 00 n.y.m.o.u.s.....
00d0: 00 00 00 00                                     ....
```

The Permissions List on this folder now has an entry for "user8" with the PidTagMemberRights property value of 0x00001800 that the client set in the previous example. The value of the PidTagMemberId property for "user8" is 0x0000001500000002 again. With that property value, the client can now send the following RopModifyPermissions request to remove the permissions for "user8" from the Permissions List:

RopModifyPermissions

RopId: 0x40

LogonID: 0

InputHandleIndex: 0 (HSOT=0x000001DA)

ModifyFlags: 0x02 IncludeFreeBusy

ModifyCount: 1

Parsing row: 1

PermissionsDataFlags: 0x02 RemoveRow

PropertyValueCount: 1 (0x01)

PidTagMemberId: 0x66710014 (PT_I8)

0x0000001500000002

```
21 bytes total in this ROP
rop(1), iLogon(1), iHsot(1), data
0000: 40 00 00 02 01 00 04 01-00 14 00 71 66 02 00 00 @..........qf...
0010: 00 15 00 00 00                                  .....
```

The server returns the following response buffer, indicating that it has successfully updated the Permissions List for the folder:

RopModifyPermissions

RopId: 0x40

InputHandleIndex: 0 (HSOT=0x000001DA)

ReturnValue: ecNone (success) (0x00000000)

```
6 bytes total in this ROP
rop(1), iHsot(1), StatusCode(4), data
0000: 40 00 00 00 00 00
```

5 Security

5.1 Security Considerations for Implementers

Implementers of this protocol have to handle the **FolderVisible**, **FolderContact**, and **FolderOwner** permissions properly. General security considerations pertaining to the underlying ROP transport protocol as specified in [MS-OXCROPS] do apply.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products:

- Microsoft® Office Outlook® 2003
- Microsoft® Exchange Server 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Exchange Server 2007
- Microsoft® Outlook® 2010
- Microsoft® Exchange Server 2010

Exceptions, if any, are noted below. If a service pack number appears with the product version, behavior changed in that service pack. 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 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 product does not follow the prescription.

<1> Section 1.7: Exchange 2007 and Exchange 2010 return a version number greater than this, and Exchange 2003 returns a version number less than this.

<2> Section 2.2.1.2.1.1: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not include the **ReplaceRows** flag in the RopModifyPermissions ROP request buffer.

<3> Section 2.2.1.5: Exchange 2003, Exchange 2007, and Exchange 2010 provide the client with the string "Anonymous" for the Anonymous Client entry and the empty string "" for the Default User entry.

<4> Section 2.2.1.6: Exchange 2007 and Exchange 2010 include the FreeBusySimple flag by default in the Calendar folder (see [MS-OXOSFLD]) for the Default User and for any non-reserved user entries in the Permissions List. It also adds the **FreeBusyDetails** to any entries that have **ReadAny** set. Exchange 2007 and Exchange 2010 use these defaults until the client modifies the Permissions List with the **IncludeFreeBusy** flag set in **ModifyFlags** to override the default value.

<5> Section 2.2.1.6: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not display the full Permissions List unless the user has FolderOwner or FolderContact permissions for the folder. If the user does not have either of those permissions, Office Outlook 2003, Office Outlook 2007, and Outlook 2010 will only display the permissions for that user on the folder by reading the PidTagRights property, as specified in [MS-OXCFOLD].

<6> Section 3.1.4.1: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do make this request, but they depend on the request failing.

<7> Section 3.1.4.2: Office Outlook 2003, Office Outlook 2007, and Outlook 2010 do not include the **ReplaceRows** flag in the RopModifyPermissions ROP request buffer.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index