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

Intellectual Property Rights Notice for Protocol Documentation

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

This document specifies the 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]:

EntryID
folder
little-endian
ROP
Unicode

The following terms are defined in the Glossary section of [MS-DTYP]:

BYTE
WORD

The following terms are specific to this document:

Anonymous Client: A client that has connected to the server without providing any user credentials <1>.

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

Permissions: Rights to access a **folder** or to perform certain operations on the **folder** based on the credentials of the user making the request.

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

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

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

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

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

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

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

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

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

[MS-XWDVSEC] Microsoft Corporation, "Web Distributed Authoring and Versioning (WebDAV) Protocol: Security Descriptor Extensions 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

None.

1.3 Protocol Overview (Synopsis)

The Exchange Access and Operation Permissions Protocol [MS-OXCPERM] is used by a client to retrieve and to set a Permissions List on a Folder [MS-OXCFOLD] stored by the server.

1.3.1 Permissions Table

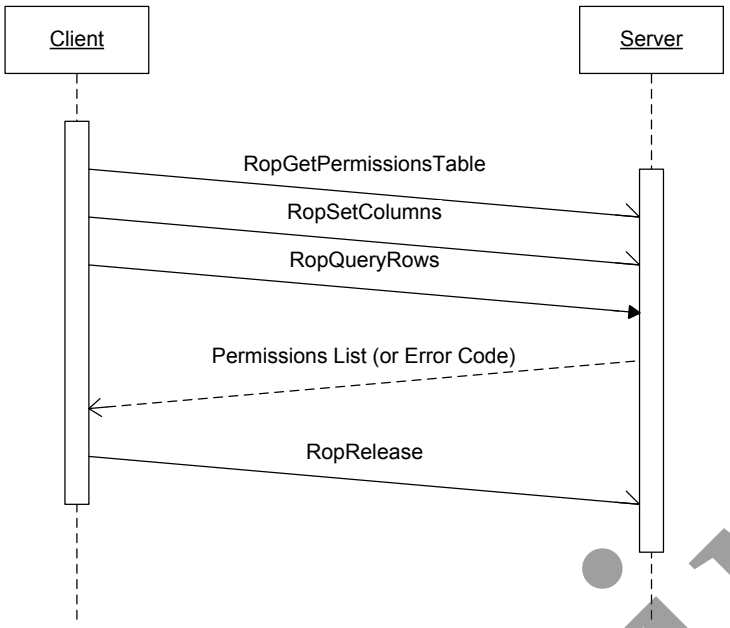

Figure 1 : Retrieving folder permissions sequence

The message sequence used to retrieve the current Permissions List is shown in Figure 1. The client sends the handle to the **Folder object** in the **RopGetPermissionsTable** message 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 containing the Permissions List for the Folder. Once the client is finished reading the Permissions List, it sends a **RopRelease** message to the server to release the Table object handle.

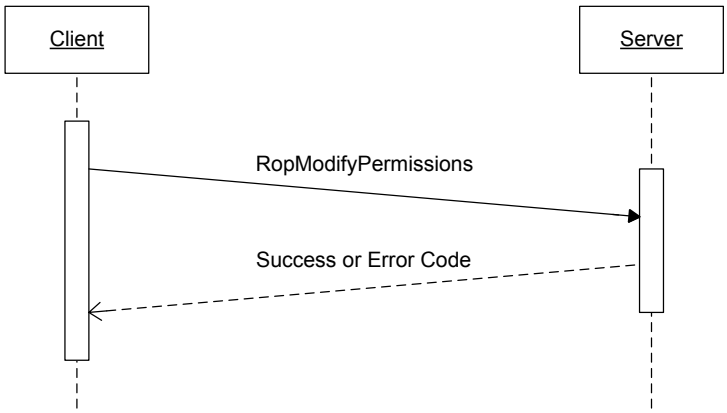

Figure 2: Setting folder permissions sequence

The message sequence used to set the access permissions is shown in Figure 2. The client builds a set of Table rows containing modifications to the Permissions List and sends them along with the handle to the Folder object to the server in the **RopModifyPermissions** message defined in section 2.2.1.2. No Table object handle is created by the server when modifying the permissions, so the client does not send a **RopRelease** message in this scenario.

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], it also extends that protocol.

This protocol depends on the Remote Operations (ROP) List and Encoding Structure Protocol [MS-OXCROPS], Table Object Protocol [MS-OXCTABL], and 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 [MS-OXCPERM] requires that the client be connected to the server 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.

1.6 Applicability Statement

A client can use the Exchange Access and Operation Permissions Protocol [MS-OXCPERM] anytime it needs to read or write the Permissions List on a folder. For instance, the client might enable another user to view a folder by adding an entry for that user to the Permissions List on the folder with read privileges.

1.7 Versioning and Capability Negotiation

Capability Negotiation: This protocol does an explicit capability negotiation as specified below 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 specified in [MS-OXOSFLD] by retrieving and setting additional Permissions which 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 greater than or equal to 8.0.360.0 <2>, the client includes the **IncludeFreeBusy** flag in the request buffer for both the **RopGetPermissionsTable** and **RopModifyPermissions** messages. The presence of the **IncludeFreeBusy** flag in the 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 using the underlying protocol specified by [MS-OXCROPS].

2.2 Message Syntax

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

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 used in this protocol.

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 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 Table Object 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:

0	1	2	3	4	5	6	7
Reserved						a	b

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

a: 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 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: 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 in 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 which are not fully documented in [MS-OXCROPS]. Other fields which are fully documented in [MS-OXCROPS] include:

- **ModifyCount**,
- **PermissionData**, and within **PermissionData**:
  - **PropertyValueCount**, and
  - **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
Reserved						a	b

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

a: 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 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: 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 <3>.

2.2.1.2.1.2 PermissionDataFlags

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

Name	Value	Description
AddRow	0x01	Adds new Permissions 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 **PropertyValue** structures specified in [MS-OXCDATA] that can be included in the **PropertyValues** field of the **PermissionData** structure 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**.

Refer to [MS-OXPROPS] and [MS-OXCDATA] for more specification details about properties, property types, and the buffer format of the **PropertyValue** structure.

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 binary BLOB which contains an entry ID 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 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 section 2.2.1.4.

2.2.1.4 PidTagMemberId

This is a PtypInteger64 value containing a unique identifier 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 returned in the response to **RopGetPermissionsTable**. The server MUST include this property in those rows.

There are 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 <4>. 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 which 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. When modifying the Permissions List with a **RopModifyPermissions** request, the server MAY NOT set the Permissions exactly as requested, depending on inter-dependencies between Permissions. 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: These bits (bitmask 0xFFFFE000) are reserved. The client and server MUST not set these flags.

a: 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: 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 <5>.

c: 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 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: 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 <6>. 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: 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: 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 use's client to create new folders within the folder.

g: 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 use's client to delete Message Objects that are owned by other users.

h: 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 use's client to modify Message Objects that are owned by other users.

i: 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 use'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: 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 use'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: 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: 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 use's client to create new Message Objects in the folder.

m: 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 use'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.

Permissions: A set of actions that a user is allowed to perform on the Folder Object or on Message Objects contained the Folder Object.

Permissions List: A list of users and the Permissions for each user on the Folder Object. This list can also include Permissions for an **Anonymous Client** and Permissions for a **Default User**.

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 MAY attempt to open a stream handle on the folder's **PidTagNtSecurityDescriptorAsXml** property, using a **RopOpenStream** request <7>. 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 of the ROP requests above 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 request buffer. The client MUST include **PropertyValue** structures in the **PermissionData** structure containing values for the following properties:

- **PidTagEntryid**
- **PidTagMemberRights**

The client MAY include the **ReplaceRows** flag in the **RopModifyPermissions** request buffer<8>, in which case all of the **PermissionData** structures in the 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 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 request buffer. The client MUST include **PropertyValue** structures in the **PermissionData** structure containing 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 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 request buffer. The client MUST include **PropertyValue** structures in the **PermissionData** structure containing values for the following properties:

- **PidTagMemberId**

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 by[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 which 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 request buffer, as specified in section 2.2.1.2.1.

3.2.5.3 Reading PidTagNtSecurityDescriptorAsXml

When the server receives a **RopOpenStream** request for the **PidTagNtSecurityDescriptorAsXml** 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 **PidTagNtSecurityDescriptorAsXml** 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:
PidTagNtSecurityDescriptorAsXml: 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, indicating that it does not support the **PidTagNtSecurityDescriptorAsXml** 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 containing 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, indicating that it 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 containing 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, indicating 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 containing 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 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 MUST be sure to enforce 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: Office/Exchange Behavior

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

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

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

---

<1> Section 1.1: The underlying ROP and RPC transports used by this protocol do not allow a client to connect to the server without providing user credentials. These permissions would only apply to clients that are using protocols other than the ROP and RPC transports to connect to the server.

<2> Section 1.7: Exchange 2007 SP1 returns a version number greater than this, and Exchange 2003 SP2 returns a version number less than this.

<3> Section 0: Neither Outlook 2003 SP3 nor Outlook 2007 SP1 includes the **ReplaceRows** flag in the **RopModifyPermissions** request buffer.

<4> Section 2.2.1.5: Exchange 2003 SP2 and Exchange 2007 SP1 provide the client with the string “Anonymous” for the **Anonymous Client** entry and the empty string “” for the **Default User** entry.

<5> Section 2.2.1.6: Exchange 2007 SP1 includes 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, and it adds the **FreeBusyDetails** to any entries that have **ReadAny** set. Exchange 2007 SP1 uses these defaults until the client modifies the Permissions List with the **IncludeFreeBusy** flag set in **ModifyFlags** to set them.

<6> Section 2.2.1.6: Outlook 2003 SP2 and Outlook 2007 SP1 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, Outlook 2003 SP2 and Outlook 2007 SP1 will only display the Permissions for that user on the folder by reading the PidTagRights property, as specified in [MS-OXCFOLD].

<7> Section 3.1.4.1: Outlook 2003 SP2 and Outlook 2007 SP1 do make this request, but they depend on the request failing.

<8> Section 3.1.4.2: Neither Outlook 2003 SP2 nor Outlook 2007 SP1 includes the **ReplaceRows** flag in the **RopModifyPermissions** request buffer.

Index