[MS-OXORULE]:
E-Mail Rules Protocol Specification

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		Updated references to reflect date of initial release.
09/03/2008	1.02		Revised and edited technical content.
12/03/2008	1.03		Minor editorial fixes.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	5.0.0	Major	Updated and revised the technical content.
05/05/2010	5.0.1	Editorial	Revised and edited the technical content.
08/04/2010	6.0	Major	Significantly changed the technical content.
11/03/2010	6.1	Minor	Clarified the meaning of the technical content.
03/18/2011	7.0	Major	Significantly changed the technical content.
08/05/2011	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/07/2011	7.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	8.0	Major	Significantly changed the technical content.
04/27/2012	9.0	Major	Significantly changed the technical content.
07/16/2012	9.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	10.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The E-Mail Rules Protocol provides the mechanism for manipulating incoming e-mail messages on a server.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

flags
GUID
handle
little-endian
remote procedure call (RPC)
Unicode

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

action
address book
binary large object (BLOB)
client-side rule
condition
contents table
Deferred Action Folder (DAF)
Deferred Action Message (DAM)
Deferred Error Message (DEM)
delegate
EntryID
extended rule
FAI contents table
folder associated information (FAI)
Folder object
hard delete
Inbox folder
Logon object
mailbox
Message object
named property
Out of Office (OOF)
Out of Office rule
property ID
property tag
public folder
recipient
remote operation (ROP)
restriction
ROP request
ROP request buffer
ROP response

ROP response buffer
rule
rules table
server-side rule
Short Message Service (SMS)
special folder
store
Table object

The following terms are specific to this document:

Rule FAI message: A folder associated information (FAI) message stored in the Inbox special folder where the client can store extra rule-related information that is opaque to the server.

rule provider: A client application that creates and maintains a specific rule. The application is identified by a unique, well-known string, which is saved as a property on the rule.

standard rule: A rule that is created, modified, or deleted by using the RopModifyRules remote operation.

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

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

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

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

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

[MS-OXCMAIL] Microsoft Corporation, "RFC2822 and MIME to E-Mail Object Conversion Algorithm".

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

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

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

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

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

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

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

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

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

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

[MS-OXWOOF] Microsoft Corporation, "Out of Office (OOF) Web Service Protocol Specification".

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

1.2.2 Informative References

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

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

[MS-OXPROTO] Microsoft Corporation, "Exchange Server Protocols System Overview".

1.3 Overview

The E-Mail Rules Protocol enables the client/server interaction that allows a messaging system to implement automatic message processing (message **rules (4)**). This protocol provides a specific mechanism through which the server and the client can implement a flexible message processing system. Mail delivery is a complex operation that allows the server and the client to implement their own additional processing that is not covered by this protocol.

Rules (4) are sets of **conditions** and associated **actions (3)** that enable a user to automatically organize, categorize, and act on messages as the messages are delivered to a folder. Rules can be set on any server folder (either **public folders** or private folders).

Rule (4) evaluation is triggered when e-mail messages are delivered in a user's **mailbox** or when messages are first saved to a public folder. The clauses in a condition in a rule (4) are evaluated against the properties of the incoming message. If the condition evaluates to "TRUE", the rule (4) actions (3) are executed either by the server or by the client. If all actions (3) in a rule (4) can be executed by the server, the rule (4) is said to be a **server-side rule**. If any action (3) cannot be executed by the server (for example, the server doesn't have access to user's personal **store**; therefore, it has to defer to the client any action (3) moving messages to a personal store), the rule (4) has to be executed by the client, and it is said to be a **client-side rule**.

Server-side rules are handled entirely by the messaging server, independent of the state of the client. Client-side rules do not execute until the client connects to the particular store on the server. For each message that needs to be acted on by the client as a result of a client-side rule, the server will create a message called **Deferred Action Message (DAM)** in a **special folder** called the **Deferred Action Folder (DAF)** as described in [MS-OXOSFLD].

All enabled rules (4) in a folder are evaluated in sequential order, one by one, until all rules (4) in the **rules table** for the particular folder have been evaluated. If the conditions of a particular rule (4) are met, its associated set of actions (3) is executed. If a rule (4) is an "exit level" rule (4) (according to a **flag** in the rule (4) state property) and the rule (4) condition is met, then the evaluation of subsequent rules (4) is canceled. Otherwise, evaluation of the next rule (4) continues even if a rule (4) action (3) moves the message, in which case the remaining rules (4) continue to run against the moved message.

If the rule (4) action is to copy or move a message to a server folder, the server will verify the existence of the destination folder. If the destination folder also has rules (4) (this is not common), the server will evaluate the destination folder rules (4) against the moved message after evaluating the remaining rules (4) in the original folder. If the destination folder does not exist, the server will create a **Deferred Error Message (DEM)** in the DAF, and the client will display an error when it processes the DEM.

When a folder is deleted, all rules (4) set on that folder are also deleted.

This protocol enables two slightly different types of rules (4): **standard rules**, which are more commonly used, and **extended rules**, which provide greater storage capacity, but for performance reasons, the server can choose to limit their usage. The way the two types of rules (4) are created and modified differs, but they are processed identically by the server and by the client.

The following subsections describe the main components covered in this protocol.

1.3.1 Creating, Modifying, and Deleting Rules

Standard rules are created, modified, and deleted by using the **remote operation (ROP)**, as described in section 2.2.1, utilizing the underlying Remote Operations (ROP) List and Encoding Protocol, as described in [MS-OXCROPS].

Extended rules are created, modified, and deleted by using a **folder associated information (FAI)** message representation as specified in section 2.2.4, using the underlying Message and Attachment Protocol, as described in [MS-OXCMSG].

1.3.2 Retrieving Rules from the Server

The client can retrieve the standard rules in a folder in the form of a **Table object**, as described in [MS-OXCTABL], by using the underlying remote operation (ROP) transport, as described in [MS-OXCROPS], in the format specified in section 2.2.2.

Each row in the returned Table object contains data representing one rule (4). The conditions, actions (3) and other rule (4) properties are returned as properties in the corresponding table row as specified in section 3.2.5.3.

To obtain a list of extended rules in a folder, the client can retrieve the **FAI contents table** for that folder. Extended rules are FAI messages identified by the value of their **PidTagMessageClass** property (section 2.2.4.1.2).

1.3.3 Executing Client-Side Rules

When a rule (4) cannot be executed entirely by the server, the client will need to complete the rule (4) execution. This is achieved via Deferred Actions, as described in section 3.1.5.1.

1.4 Relationship to Other Protocols

This protocol is dependent on the protocols related folders, messages, and tables, as described in [MS-OXCFOLD], [MS-OXCMSG], and [MS-OXCTABL]. The protocol also relies on utilizing ROPs transmitted to the server using the underlying transport, as described in [MS-OXCROPS].

Extended rules use **Message objects** described in [MS-OXCMSG] as an underlying transport.

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

1.5 Prerequisites/Preconditions

This protocol assumes the client has previously logged on to the messaging server as described in [MS-OXCROPS] and has acquired a **handle** to the folder it needs to set the rules (4) to and retrieve the rules (4) from, as described in [MS-OXCFOLD]. This protocol also relies on the use of the underlying ROP transport protocol described in [MS-OXCROPS].

1.6 Applicability Statement

This protocol can be used to build automatic workflows for messages that are delivered by the server into a message folder.

1.7 Versioning and Capability Negotiation

This protocol defines version 1 of the Extended Rule Actions format, as specified in section 2.2.4.2.

1.8 Vendor-Extensible Fields

A third party application can create its own set of rules (4) by using its custom string as the value of the **PidTagRuleProvider** property as specified in section 2.2.1.3.1.5. There is no centralized authority that ensures uniqueness of **rule provider** strings across different client applications.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The standard rules, as specified in sections 2.2.1, 2.2.2, and 2.2.3, are built by using the ROP List and Encoding Protocol specified in [MS-OXCROPS]. The extended rules portion of the protocol, as specified in section 2.2.4, is built by using the Message and Attachment Protocol specified in [MS-OXCMSG].

The **ROP request buffer** and **ROP response buffer** specified by this protocol are sent to and received from the server respectively using the underlying protocol specified in [MS-OXCROPS].

2.2 Message Syntax

Standard rules are the most common and typical way of specifying rules (4) for a folder. Sections 2.2.1, 2.2.2, and 2.2.3 specify the ROP request buffers and ROP response buffers specific to this protocol. The syntax of these requests and responses is documented in [MS-OXCROPS], as specified in each section below.

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 in **little-endian** format.

2.2.1 RopModifyRules ROP

The **RopModifyRules** ROP ([MS-OXCROPS] section 2.2.11.1) creates, modifies, or deletes rules (4) in a folder.

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

2.2.1.1 RopModifyRules ROP Request Buffer

The following descriptions define valid fields for the **RopModifyRules** ROP request buffer ([MS-OXCROPS] section 2.2.11.1.1).

InputHandleIndex (8 bits): The index to the input handle for this operation, which is a **Folder object** handle representing the folder for which rules (4) are to be modified.

ModifyRulesFlag (8 bits): A bitmask that specifies how the rules (4) included in this structure are created on the server. Its structure is as follows.

0	1	2	3	4	5	6	7
x	x	x	x	x	x	x	R

R (Bitmask 0x01): If this bit is set, the rules (4) in this request are to replace the existing set of rules (4) in the folder; in this case, all subsequent **RuleData** structures, as specified in section 2.2.1.3, MUST have the **ROW_ADD** flag as the value of their **RuleDataFlags** field, as specified in section 2.2.1.3.1. If this bit is not set, the rules (4) specified in this request represent changes (delete, modify, and add) to the set of rules (4) already existing in this folder.

x: This bit MUST be set to zero (0) when sent and MUST be ignored when received.

RulesCount (2 bytes): The value of the **RulesCount** field is equal to the number of **RuleData** structures present in the ROP request buffer.

RulesData (variable): An array of **RuleData** structures that use the format specified in section 2.2.1.3.

2.2.1.2 RopModifyRules ROP Response Buffer

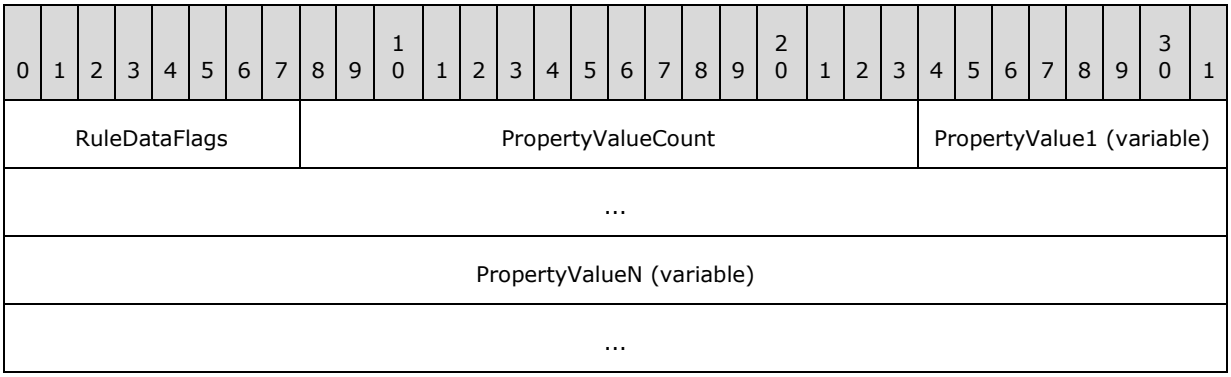
The following descriptions define valid fields for the **RopModifyRules** ROP response buffer ([MS-OXCROPS] section 2.2.11.1.2).

InputHandleIndex (8 bits): The input handle in the response buffer MUST be the same as the index to the input handle in the request buffer for this operation.

ReturnValue (4 bytes): A value that indicates the result of the operation. To indicate success, the server returns 0x00000000. For a list of common error return values, see [MS-OXCDATA] section 2.4.

2.2.1.3 RuleData Structure

The **RopModifyRules** ROP request buffer ([MS-OXCROPS] section 2.2.11.1) MUST contain the number of **RuleData** structure buffers equal to the value of the **RulesCount** field, as specified in [MS-OXCROPS] section 2.2.11.1.1.1. The format of the **RuleData** structures in the **RopModifyRules** ROP request buffer is specified as follows.



RuleDataFlags (1 byte): Contains flags specifying whether the rule (4) is to be added, modified, or deleted according to the following values.

Flag name	Value	Description
ROW_ADD	0x01	Adds the data in the rule buffer to the rule set as a new rule (4).
ROW_MODIFY	0x02	Modifies the existing rule (4) identified by the value of the **PidTagRuleId** property (section 2.2.1.3.1.1).
ROW_REMOVE	0x04	Removes from the rule set the rule (4) that has the same value of the **PidTagRuleId** property.

PropertyValueCount (2 bytes): The count of properties that are defined in this structure. This field MUST be greater than zero and MUST be followed by a number of **TaggedPropertyValue** structures equal to the value of the **PropertyValueCount** field.

PropertyValue1 (variable): A **TaggedPropertyValue** structure ([MS-OXCDATA] section 2.11.4) containing one **property tag** and its associated value. The property tag used in this buffer MUST be among the ones specified in section 2.2.1.3.1.

PropertyValueN (variable): Last of the **PropertyValueCount TaggedPropertyValue** structures.

2.2.1.3.1 PropertyValue Structure

The rules (4)-related property tags to be used in the **TaggedPropertyValue** structure are specified in section 2.2.1.3.1.1 through section 2.2.1.3.1.10. The format of the **TaggedPropertyValue** structure is specified in [MS-OXCDATA] section 2.11.4.

2.2.1.3.1.1 PidTagRuleId Property

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

The **PidTagRuleId** property ([MS-OXPROPS] section 2.1013) specifies a unique identifier the messaging server generates for each rule (4) when the rule (4) is first created. The **PidTagRuleId** property MUST NOT be used when requesting that a new rule (4) be created but MUST be used when requesting that a rule (4) be modified or deleted.

2.2.1.3.1.2 PidTagRuleSequence Property

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

The **PidTagRuleSequence** property ([MS-OXPROPS] section 2.1026) contains a value used to determine the order in which rules (4) are evaluated and executed. Rules (4) are evaluated in sequence according to the increasing order of this value. The evaluation order for rules (4) that have the same value in the **PidTagRuleSequence** property is undefined: the server can choose an arbitrary order for rules (4) with the same value, but that does not affect the sequence of other rules (4).

2.2.1.3.1.3 PidTagRuleState Property

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

The **PidTagRuleState** property ([MS-OXPROPS] section 2.1027) contains a value interpreted as a Bitmask combination of flags that specify the state of the rule (4). The value of the **PidTagRuleState** property is defined as follows.

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
x	PE	SCL	EL	HI	OF	ER	EN	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

EN (ST_ENABLED, Bitmask 0x00000001): The rule (4) is enabled for execution. If neither this flag nor the **ST_ONLY_WHEN_OOF** flag are set, the server skips this rule (4) when evaluating rules (4).

ER (ST_ERROR, Bitmask 0x00000002): The server has encountered any nonparsing error processing the rule (4). This flag is not to be set by the client and is to be ignored by the server if it is.

OF (ST_ONLY_WHEN_OOF, Bitmask 0x00000004): The rule (4) is executed only when a user sets the **Out of Office (OOF)** state on the mailbox, as specified in [MS-OXWOOF] section 2.2.5.2. This flag MUST NOT be set in a public folder rule (4). For details on this flag, see section 3.2.5.1.1.1.

HI (ST_KEEP_OOF_HIST, Bitmask 0x00000008): For details, see section 3.2.5.1.1. This flag MUST NOT be set in a public folder rule (4).

EL (ST_EXIT_LEVEL, Bitmask 0x00000010): Rule (4) evaluation will terminate after executing this rule (4), except for evaluation of **Out of Office rules**. For details, see section 3.2.5.1.1.1.

SCL (ST_SKIP_IF_SCL_IS_SAFE, Bitmask 0x00000020): Evaluation of this rule (4) will be skipped if the delivered message's **PidTagContentFilterSpamConfidenceLevel** property ([MS-OXPROPS] section 2.717) has a value of 0xFFFFFFFF.

PE (ST_RULE_PARSE_ERROR, Bitmask 0x00000040): The server has encountered rule (4) data from the client that is in an incorrect format, which caused an error parsing the rule (4) data. This flag is not to be set by the client and is to be ignored by the server if it is.

x: Unused by this protocol. Bit locations marked with x are to be set to 0, SHOULD NOT be modified by the client, and are ignored by the server.<1>

2.2.1.3.1.4 PidTagRuleName Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleName** property ([MS-OXPROPS] section 2.1023) specifies the name of the rule (4).

2.2.1.3.1.5 PidTagRuleProvider Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleProvider** property ([MS-OXPROPS] section 2.1024) identifies the client application that owns the rule (4). The client specifies this property when adding or modifying a rule (4).

Rules that are stored on folders are associated with the application that owns the rules (4) by using a rule provider string. Each client application is to only add, modify or delete rules (4) that it is responsible for.

A client can define its own rule provider string. The value of the string MUST NOT be the same as a rule provider string being used by another client that could be setting rules (4) on the same folder.<2>

2.2.1.3.1.6 PidTagRuleLevel Property

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

The **PidTagRuleLevel** property ([MS-OXPROPS] section 2.1015) is not used; if a client requests that this property be set, the requested value MUST be 0x00000000.

2.2.1.3.1.7 PidTagRuleUserFlags Property

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

The **PidTagRuleUserFlags** property ([MS-OXPROPS] section 2.1028) is an opaque property that the client sets for the exclusive use of the client. The server is to preserve this value if set by the client but ignores its contents during rule (4) evaluation and processing.

2.2.1.3.1.8 PidTagRuleProviderData Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleProviderData** property ([MS-OXPROPS] section 2.1025) is an opaque property that the client sets for the exclusive use of the client. The server is to preserve this value if set by the client but ignores its contents during rule (4) evaluation and processing.

2.2.1.3.1.9 PidTagRuleCondition Property

Type: **PtypRestriction** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleCondition** property ([MS-OXPROPS] section 2.1010) sets the condition used when evaluating the rule (4). The condition is expressed as a **restriction (2)**, as specified in [MS-OXCDATA] section 2.12, and the **PropertyValue** buffer contains the restriction (2) structure packaged as specified in [MS-OXCDATA] (using **WORD** ([MS-DTYP]) values).

2.2.1.3.1.10 PidTagRuleActions Property

Type: **PtypRuleAction** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleActions** property ([MS-OXPROPS] section 2.1008) contains the set of actions (3) associated with the rule (4). Its structure is specified in section 2.2.5, using a **WORD** ([MS-DTYP]) value.

2.2.2 RopGetRulesTable ROP

The **RopGetRulesTable** ROP ([MS-OXCROPS] section 2.2.11.2) creates a Table object through which the client can access the standard rules in a folder using table operations as specified in [MS-OXCTABL]. The table returned by the server is required to contain all standard rules associated with a given folder. Each row in the table MUST represent one rule (4).

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

2.2.2.1 RopGetRulesTable ROP Request Buffer

The following descriptions define valid fields for the **RopGetRulesTable** ROP request buffer ([MS-OXCROPS] section 2.2.11.2.1).

InputHandleIndex (8 bits): The index to the input handle for this operation, which is a Folder object handle representing the folder for which rules (4) are to be retrieved.

TableFlags (8 bits): The possible values for these bits are as follows.

0	1	2	3	4	5	6	7
x	x	x	x	x	x	x	U

U (Bitmask 0x40): This bit is set if the client is requesting that string values in the table be returned as **Unicode** strings.

x: These unused bits MUST be set to zero (0) by the client. The server SHOULD<3> return an error if these bits are nonzero but can ignore them.

2.2.2.2 RopGetRulesTable ROP Response Buffer

The following descriptions define valid fields for the **RopGetRulesTable** ROP response buffer ([MS-OXCROPS] section 2.2.11.2.2).

OutputHandleIndex (8 bits): The index to the output handle for this operation. MUST be set to the value of the **OutputHandleIndex** field specified in the request.

ReturnValue (4 bytes): An integer indicatingthe result of the operation. To indicate success, the server returns 0x00000000. For a list of common error return values, see [MS-OXCDATA] section 2.4.

2.2.3 RopUpdateDeferredActionMessages ROP

The **RopUpdateDeferredActionMessages** ROP ([MS-OXCROPS] section 2.2.11.3) instructs the server to update the **PidTagDamOriginalEntryId** property (section 2.2.6.3) on one or more DAMs.

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

2.2.3.1 RopUpdateDeferredActionMessages ROP Request Buffer

The following descriptions define valid fields for the **RopUpdateDeferredActionMessages** ROP request buffer ([MS-OXCROPS] section 2.2.11.3.1).

InputHandleIndex (8 bits): The index to the input handle for this operation, which is a **Logon object** handle.

ServerEntryIdSize (2 bytes): The length, in bytes, of the **ServerEntryId** field.

ServerEntryId (variable): A byte array representing the **EntryID** of the DAM on the server. The length of this byte array is specified by the **ServerEntryIdSize**field.

ClientEntryIdSize (2 bytes): A **WORD** value representing the length, in bytes, of the **ClientEntryId** field.

ClientEntryId (variable): A byte array representing the EntryID of the message downloaded by the client to which the DAM will now apply. The length of this byte array is specified by the **ClientEntryIdSize** field.

2.2.3.2 RopUpdateDeferredActionMessages ROP Response Buffer

The following descriptions define valid fields for the **RopUpdateDeferredActionMessages** ROP response buffer ([MS-OXCROPS] section 2.2.11.3.2).

InputHandleIndex (8 bits): The index to the input handle for this operation. This value MUST be the same as the index to the input handle in the request buffer for this operation.

ReturnValue (4 bytes): The result of the operation. To indicate success, the server returns 0x00000000. For a list of common error return values, see [MS-OXCDATA] section 2.4.

2.2.4 Extended Rules Message Syntax

Using standard rules for message processing, as specified in section 2.2.1, section 2.2.2, and section 2.2.3, has one major limitation as a consequence of using the ROP layer as the underlying transport: there is an inherent size limitation of 32 kilobytes per ROP package. To work around this limitation, extended rules were created. Extended rules are built using the Message and Attachment Protocol as specified in [MS-OXCMSG], so that messages can be spread over multiple ROPs to avoid the size limitation. An extended rule is defined as an FAI message in a folder that has the value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) set to "IPM.ExtendedRule.Message". This FAI message also has a set of rule-related properties set on it, as specified in the following subsections. To create, modify, or delete an extended rule, the application is required to create, modify, or delete the underlying FAI message.

Extended rules use a different set of properties than the **RopModifyRules** ROP ([MS-OXCROPS] section 2.2.11.1). However, these properties map to properties for **RopModifyRules**; and except where noted, their formats are identical and the same syntactic restrictions (2) and semantic meanings of values apply as the respective property defined in section 2.2.1.3.1.

2.2.4.1 Properties of an Extended Rule

The following properties have a particular meaning when set on FAI messages representing an extended rule. The application can store additional meta-data in any other property on the FAI message. The server is to ignore any properties not explicitly listed here when evaluating an extended rule.

2.2.4.1.1 PidTagRuleMessageName Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleMessageName** property ([MS-OXPROPS] section 2.1017) SHOULD be set on the FAI message. This property has the same semantics as the **PidTagRuleName** property (section 2.2.1.3.1.4).

2.2.4.1.2 PidTagMessageClass Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) MUST be set on the FAI message and MUST have a value of "IPM.ExtendedRule.Message".

2.2.4.1.3 PidTagRuleMessageSequence Property

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

The **PidTagRuleMessageSequence** property ([MS-OXPROPS] section 2.1020) MUST be set on the FAI message. This property has the same semantics as the **PidTagRuleSequence** property (section 2.2.1.3.1.2).

2.2.4.1.4 PidTagRuleMessageState Property

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

The **PidTagRuleMessageState** property ([MS-OXPROPS] section 2.1021) MUST be set on the FAI message. This property has the same semantics and flag meanings as the **PidTagRuleState** property (section 2.2.1.3.1.3).

2.2.4.1.5 PidTagRuleMessageUserFlags Property

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

This **PidTagRuleMessageUserFlags** property ([MS-OXPROPS] section 2.1022) MAY be set on the FAI message. This property has the same semantics as the **PidTagRuleUserFlags** property (section 2.2.1.3.1.7).

2.2.4.1.6 PidTagRuleMessageLevel Property

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

The **PidTagRuleMessageLevel** property ([MS-OXPROPS] section 2.1016) SHOULD be set on the FAI message. This property has the same semantics as the **PidTagRuleLevel** property (section 2.2.1.3.1.6).

2.2.4.1.7 PidTagRuleMessageProvider Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleMessageProvider** property ([MS-OXPROPS] section 2.1018) MUST be set on the FAI message. This property has the same semantics as the **PidTagRuleProvider** property (section 2.2.1.3.1.5).

2.2.4.1.8 PidTagRuleMessageProviderData Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleMessageProviderData** property ([MS-OXPROPS] section 2.1019) MAY be set on the FAI message. This property has the same syntax and semantics as the **PidTagRuleProviderData** property (section 2.2.1.3.1.8).

2.2.4.1.9 PidTagExtendedRuleMessageActions Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagExtendedRuleMessageActions** property ([MS-OXPROPS] section 2.761) MUST be set on the FAI message. This property serves the same purpose as the **PidTagRuleActions** property (section 2.2.1.3.1.10); however, it contains additional information about the **named properties** used. All string values contained in any part of the action buffer used to contain actions (3) MUST be in Unicode format. The format of the **PidTagExtendedRuleMessageActions** property is defined in section 2.2.4.2.

2.2.4.1.10 PidTagExtendedRuleMessageCondition Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagExtendedRuleMessageCondition** property ([MS-OXPROPS] section 2.762) MUST be set on the FAI message. This property serves the same purpose as the **PidTagRuleCondition** property (section 2.2.1.3.1.9); however, it contains additional information about the named properties used.

All string values contained in any part of this condition property value MUST be in Unicode format. The format of this property is defined in section 2.2.4.3. If the **PidTagExtendedRuleSizeLimit** property is set on the Logon object (as specified in [MS-OXCSTOR] section 2.2.2.1), the client is required to keep the size of the **PidTagExtendedRuleMessageCondition** property under the value specified by the value of the **PidTagExtendedRuleSizeLimit** property; conversely, the server is required to return an error if the client attempts to set a binary property whose size is above the value specified by the value of the **PidTagExtendedRuleSizeLimit** property.

For a list of common error return values, see [MS-OXCDATA] section 2.4.

2.2.4.2 Extended Rule Actions Format

An extended rule's **PidTagExtendedRuleMessageActions** property (section 2.2.4.1.9) contains additional information about the version of the rule (4) and the named properties stored in the rule (4) action (3), as well as information about the actions (3) to be performed by this rule (4). The format of the binary property is specified as follows.

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
NamedPropertyInformation (variable)																															
...																															
RuleVersion																															
RuleActionsBuffer (variable)																															
...																															

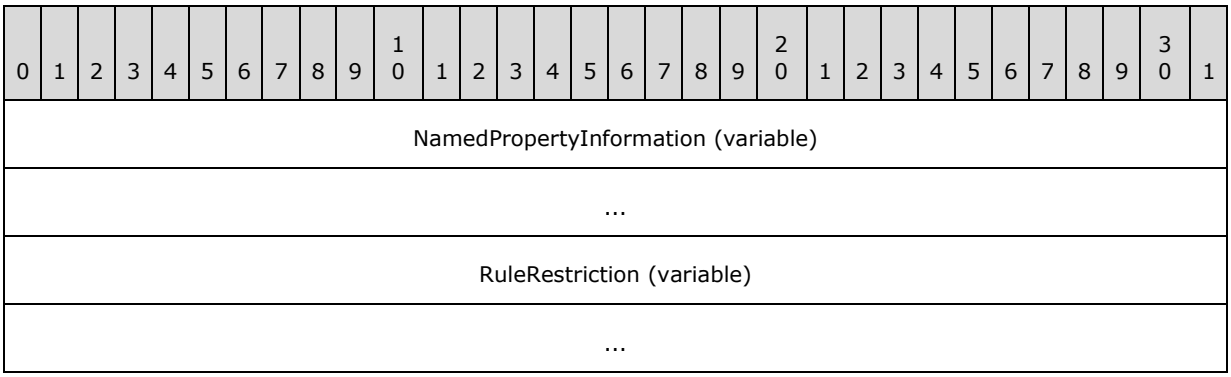
NamedPropertyInformation (variable): Specifies information about named properties used in this action (3) as specified in section 2.2.4.4.

RuleVersion (4 bytes): Specifies the extended rules version format. This document defines version 1, and thus this value MUST be set to 0x00000001.

RuleActionsBuffer (variable): A structure containing the actions (3) to be executed when the rule condition for the rule (4) to which these actions (3) apply evaluates to "TRUE". The format of this structure is defined in section 2.2.5, using a **DWORD** ([MS-DTYP]) value.

2.2.4.3 Extended Rule Condition Format

Similar to extended rule actions (3), extended rule conditions contain information about any named properties contained inside of them. The format of the **PidTagExtendedRuleMessageCondition** binary property (section 2.2.4.1.10) is specified as follows.



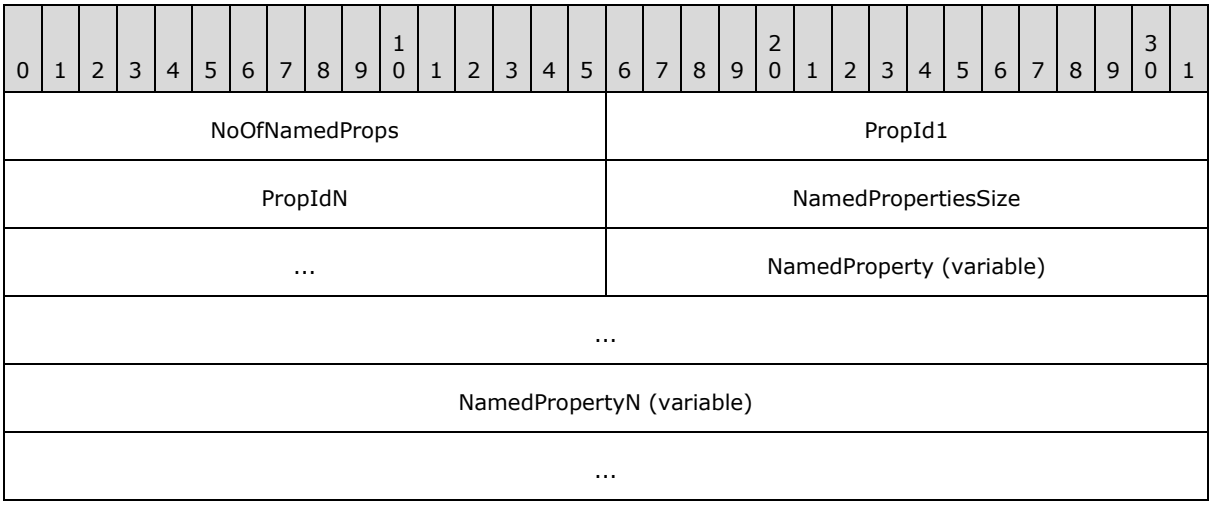
NamedPropertyInformation (variable): Specifies information about named properties used in this condition, as specified in section 2.2.4.4.

RuleRestriction (variable): A structure containing the condition to be evaluated, represented as a **Restriction** structure. The format of this **Restriction** structure is defined in [MS-OXCDATA] section 2.12, using a **DWORD** [MS-DTYP] value.

2.2.4.4 Named Property Information Format

The named property information format provides context to any named property tags that are present in the structure it precedes. For every distinct (unique) named property used in the structure it precedes, the **Named Property Information** structure contains one **PropId** – **NamedProperty** pair. Each **PropId** field is a **property ID** with a value of 0x8000 or greater and uniquely identifies the named property within an extended rule.

The format of the **Named Property Information** structure is specified as follows.



NoOfNamedProps (2 bytes): Specifies the number of named property mappings that are packed in this buffer. If no named properties are used in the structure that follows the **Named Property Information** buffer, the value of this field MUST be 0x0000.

PropId1 (2 bytes): The first **PropId** field.

PropIdN (2 bytes): The last (**NoOfNamedProps**) **PropId** field.

NamedPropertiesSize (4 bytes): The total size, in bytes, of the following fields. Only present if **NoOfNamedProps** is greater than zero.

NamedProperty (variable): Specifies the first **PropertyName** structure, whose format is specified in [MS-OXCDATA] section 2.6.1.

NamedPropertyN (variable): Specifies the last (**NoOfNamedProps**) **PropertyName** structure.

Note that if there are no named properties to be listed, the **Named Property Information** structure reduces to a 2-byte **WORD** value of 0x0000.

2.2.5 Rule Action Format

The rule action data buffer MUST have one or more blocks of a binary data to specify various actions (3) of the rule (4), as 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
NoOfActions (2-bytes or 4-bytes)																															
ActionBlock1 (variable)																															
...																															
ActionBlockN (variable)																															

NoOfActions (4 bytes): Specifies the number of **ActionBlocks** that are packed in this buffer. This number MUST be greater than zero.

ActionBlock1 (variable): Specifies an action (3); see section 2.2.5.1.

ActionBlockN (variable): Specifies an action (3) (the last of the **NoOfActions ActionBlock** fields).

2.2.5.1 Action Block Buffer Format

The format of an action data block buffer is specified as follows.

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
ActionLength (4 bytes)																															
ActionType																ActionFlavor															
...																ActionFlags															
...																ActionData (variable)															

ActionLength (4 bytes): Contains the cumulative length in bytes of the subsequent fields in this **ActionBlock**.

ActionType (1 byte): Specifies the types of action (3) (see table in section 2.2.5.1.1).

ActionFlavor (4 bytes): MUST be used in conjunction with specific **ActionTypes** that support it, and MUST be zero otherwise (see section 2.2.5.1.2).

ActionFlags (4 bytes): Client-defined flags. The **ActionFlags** field is used solely by the client.<4> It is not used by the server but stored only.

ActionData (variable): Specifies action (3) data based on the **ActionType**. For more details, see section 2.2.5.1.3.

2.2.5.1.1 Action Types

The **ActionType** field MUST have one of the following values.

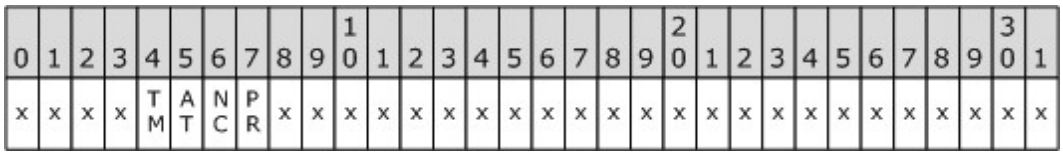
Action name	Value	Meaning
OP_MOVE	0x01	Moves the message to a folder. MUST NOT be used in a public folder rule (4).
OP_COPY	0x02	Copies the message to a folder. MUST NOT be used in a public folder rule (4).
OP_REPLY	0x03	Replies to the message.
OP_OOF_REPLY	0x04	Sends an OOF reply to the message.
OP_DEFER_ACTION	0x05	Used for actions (3) that cannot be executed by the server (like playing a sound). MUST NOT be used in a public folder rule (4).
OP_BOUNCE	0x06	Rejects the message back to the sender.
OP_FORWARD	0x07	Forwards the message to a **recipient (2)** address.
OP_DELEGATE	0x08	Assigns the message to another recipient (2).
OP_TAG	0x09	Adds or changes a property on the message.
OP_DELETE	0x0A	Deletes the message.
OP_MARK_AS_READ	0x0B	Sets the MSGFLAG_READ flag in the **PidTagMessageFlags** property ([MS-OXCMSG] section 2.2.1.6) on the message.

2.2.5.1.2 Action Flavors

The **ActionFlavor** field contains flags used in conjunction with the **ActionType** field and specifies additional information associated with the action (3) to be taken.

The only **ActionType** field values that currently support an Action Flavor are "OP_REPLY", "OP_OOF_REPLY" and "OP_FORWARD". The value of the **ActionFlavor** field MUST be 0x00000000 if the value of the **ActionType** field is not one of these values.

If the value of the **ActionType** field is "OP_FORWARD", the **ActionFlavor** field contains a combination of the bitwise flags specified as follows.



PR (Bitmask 0x00000001): Preserves the sender information and indicates that the message was autoforwarded. Can be combined with the NC **ActionFlavor** flag.

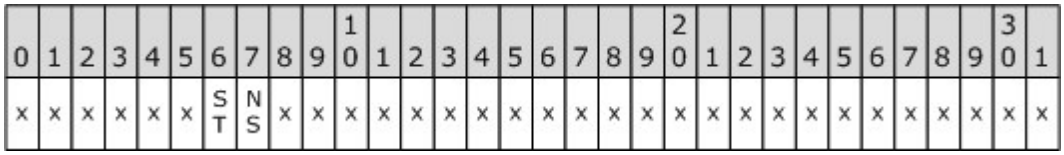
NC (Bitmask 0x00000002): Forwards the message without making any changes to the message. Can be combined with the PR **ActionFlavor** flag.

AT (Bitmask 0x00000004): Makes the message an attachment to the forwarded message. This value MUST NOT be combined with other **ActionFlavor** flags.

TM (Bitmask 0x00000008): Indicates that the message SHOULD<5> be forwarded as a **Short Message Service (SMS)** text message. This value MUST NOT be combined with other **ActionFlavor** flags.

x: Unused. This bit MUST be set to 0 by the client and ignored by the server.

If the **ActionType** field value is "OP_REPLY" or "OP_OOF_REPLY", the **ActionFlavor** field MUST have one of the values specified in the following table or zero (0x00000000). (A value of zero (0x00000000) indicates standard reply behavior, as specified in section 3.1.4.2.5.)



NS (Bitmask 0x00000001): Do not send the message to the message sender (the reply template MUST contain recipients (2) in this case).

ST (Bitmask 0x00000002): Server will use fixed, server-defined text in the reply message and ignore the text in the reply template. This text is an implementation detail.

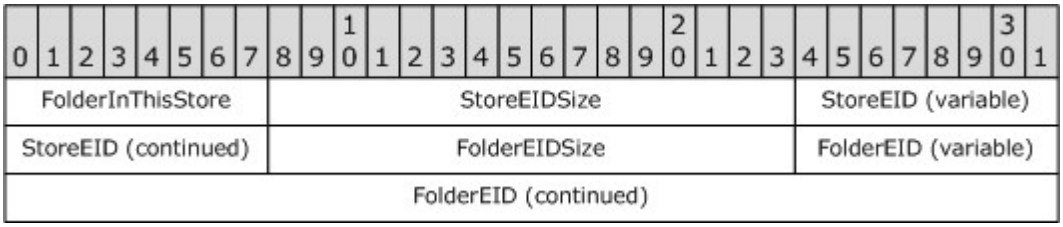
x: Unused. This bit MUST be set to 0 by the client and ignored by the server.

2.2.5.1.3 Action Data Buffer Format

The **ActionData** buffer is different for each **ActionType** field and MUST use the appropriate format specified in this section.

2.2.5.1.3.1 OP_MOVE and OP_COPY Action Data Buffer Format

A Move/Copy action (3) is used to move or copy an incoming message to a specified folder in the destination store. The **ActionData** buffer used in an action (3) of type "OP_MOVE" or "OP_COPY" and the packet structure is specified as follows.



FolderInThisStore (1 byte): Indicates whether the folder is in the server store. MUST be either 0x01 if the folder whose EntryID is **FolderEID** is in the server store, or 0x00 if the folder is in a different store (for example, a local store the server cannot access).

StoreEIDSize (2 bytes): The size of the **StoreEID** byte array.

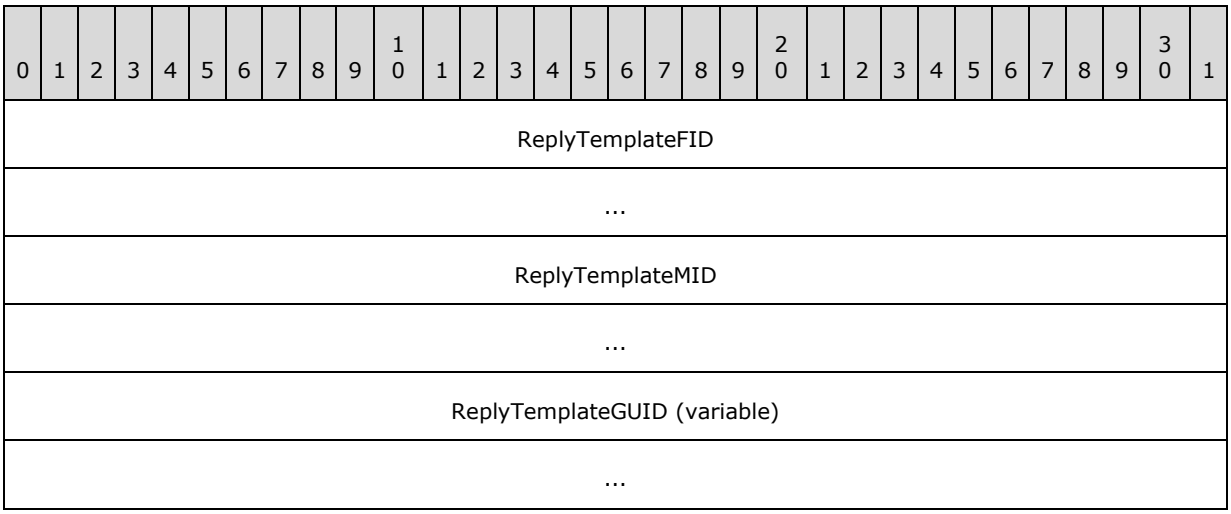
StoreEID (variable): The binary buffer specifies the destination store EntryID. The **StoreEID** byte array is specified in [MS-OXCDATA] section 2.2.4.3.

FolderEIDSize (2 bytes): The size of the **FolderEID** byte array.

FolderEID (variable): The binary buffer specifies the destination folder's EntryID. If the value of the **FolderInThisStore** field is 0x01, the structure of this field is that specified in [MS-OXCDATA] section 2.2.4.1. If the value of the **FolderInThisStore** field is 0x00, the structure of this field is undefined (that is, this field is an opaque **binary large object (BLOB)**).

2.2.5.1.3.2 OP_REPLY and OP_OOF_REPLY Action Data Buffer Format

The Reply/OOF Reply **ActionData** buffer format is specified as follows.



ReplyTemplateFID (8 bytes): The Reply template folder ID (FID), as specified in [MS-OXCDATA] section 2.2.1.1. For details about creating a Reply template, see section 3.1.4.2.5.

ReplyTemplateMID (8 bytes): The Reply template message ID (MID), as specified in [MS-OXCDATA] section 2.2.1.2.

ReplyTemplateGUID (variable): The **ReplyTemplateGUID** field in the Reply **ActionData** buffer is the value of the **GUID** generated by the client in the process of creating a Reply template. The value of the **ReplyTemplateGUID** field is also stored on the Reply template

message as the value of the **PidTagReplyTemplateId** property (section 2.2.9.2). The value of the **ReplyTemplateGUID** field MUST be unique in the folder - no two Reply templates can share the same GUID. Before creating a rule (4) that has a value of "OP_REPLY" or "OP_OOF_REPLY" in the **ActionType** field, the client is required to first create a Reply template FAI message in the same folder as the rule (4). For more details about working with FAI messages, see [MS-OXCFOLD] and [MS-OXCMSG].

2.2.5.1.3.3 OP_DEFER_ACTION Action Data Buffer Format

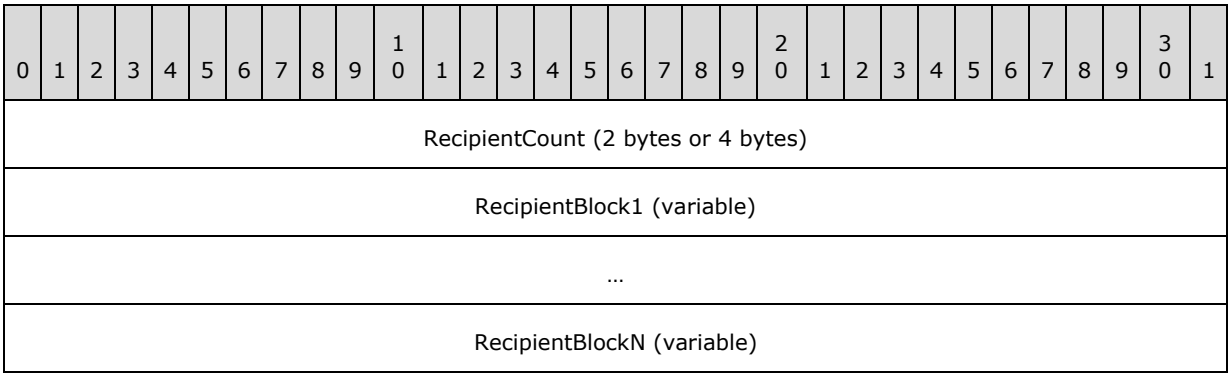
If one or more actions (3) for a specific rule (4) cannot be executed on the server, the rule (4) is required to be a client-side rule, with a value in the **ActionType** field of "OP_DEFER_ACTION". Execution of the rule (4) is postponed until the client is available.

The client encodes the rule (4) information as a client-dependent data structure designating the action (3) to be performed. The format is client-implementation-dependent and contains enough information to allow the client to perform the client-side operation when requested. The size of the buffer is obtained by reading the value in the **ActionLength** field in the **ActionBlock** structure containing an **OP_DEFER_ACTION** in the **ActionType** field.

If the action (3) type is "OP_DEFER_ACTION", the **ActionData** buffer is completely under the control of the client that created the rule (4). This binary buffer MUST be treated as an opaque BLOB by the server. When a message that satisfies the rule (4) condition is received, the server creates a DAM and places the entire content of the **ActionBlock** field as part of the **PidTagClientActions** property (section 2.2.6.6) on the DAM as specified in sections 3.2.5.1.2, 2.2.6, and 2.2.6.6.

2.2.5.1.3.4 OP_FORWARD and OP_DELEGATE Action Data Buffer Format

The **ActionData** buffer format that MUST be used with the "OP_FORWARD" and "OP_DELEGATE" action (3) types is specified as follows.



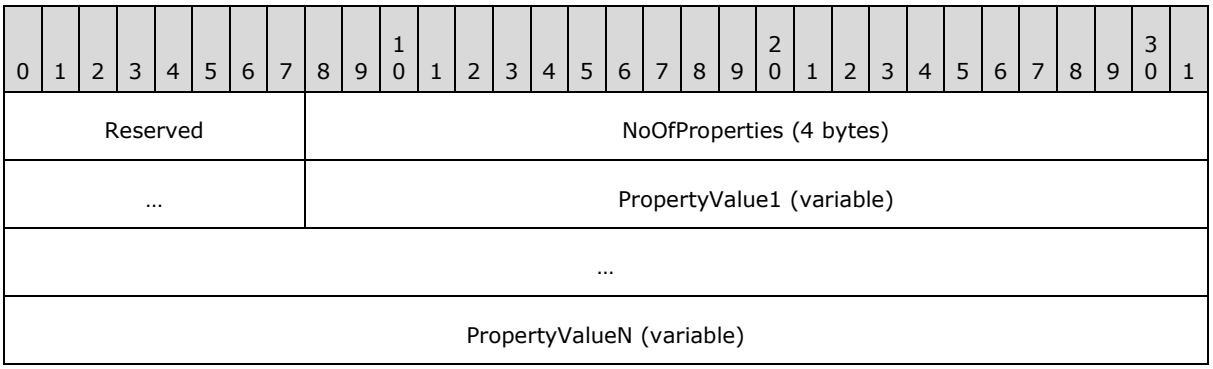
RecipientCount (4 bytes): Specifies the number of recipient (2) blocks. This number MUST be greater than zero.

RecipientBlock1 (variable): Specifies recipient (2) information. The **RecipientBlock** data buffer packet structure is specified in section 2.2.5.1.3.4.1.

RecipientBlockN (variable): Last of **RecipientCount RecipientBlocks**.

2.2.5.1.3.4.1 RecipientBlock Data Buffer Packet Structure

The **RecipientBlock** data buffer packet structure is specified as follows.



Reserved (1 byte): This value is implementation-specific and not required for interoperability.<6>

NoOfProperties (4 bytes): Specifies the number of properties in the block. This number MUST be greater than zero.

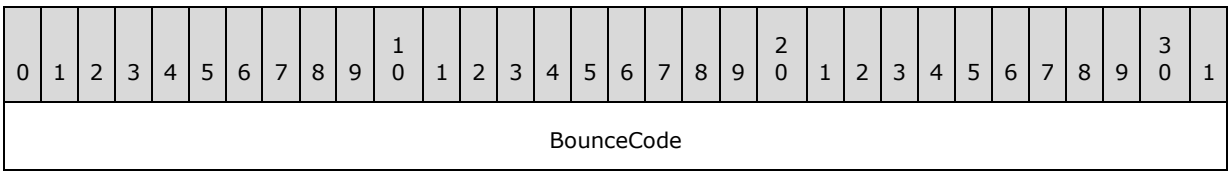
PropertyValue1 (variable): Specifies the first **TaggedPropertyValue** structure, as specified in [MS-OXCDATA] section 2.11.4.

PropertyValueN (variable): Last of **NoOfProperties TaggedPropertyValue** structures.

The client is required to, at a minimum, specify values for the **PidTagDisplayName** ([MS-OXCFOLD] section 2.2.2.2.2.4), **PidTagEmailAddress** ([MS-OXOABK] section 2.2.3.14), and **PidTagRecipientType** ([MS-OXOMSG] section 2.2.3.1) properties in the forward/**delegate ActionData** buffer, as specified in section 2.2.5.1.3.4; some rules (4) MAY<7> require more.

2.2.5.1.3.5 OP_BOUNCE Action Data Buffer Format

The Bounce **ActionData** buffer format packet structure is specified as follows.



BounceCode (4 bytes): Specifies a bounce code.

The **BounceCode** field MUST have one of the following values.

Value	Meaning
0x0000000D	The message was rejected because it was too large.
0x0000001F	The message was rejected because it cannot be displayed to the user.
0x00000026	The message delivery was denied for other reasons.

2.2.5.1.3.6 OP_TAG Action Data Buffer Format

An "OP_TAG" action Data Buffer is a **TaggedPropertyValue** structure, packaged as specified in [MS-OXCDATA] section 2.11.4.

2.2.5.1.3.7 OP_DELETE or OP_MARK_AS_READ Data Buffer Format

For the **OP_DELETE** or **OP_MARK_AS_READ** action types, the incoming messages are deleted<8> or marked as read according to the **ActionType** itself. These actions (3) have no **ActionData** buffer.

2.2.6 DAM Syntax

A DAM has to be created by the server to indicate to the client that it needs to further process a client-side rule action (3). This process is specified in section 3.2.5.1.2. Extended rules are not used in DAMs.

In addition to properties required on any message (as specified in [MS-OXCMSG] section 2.2.1), the following properties are specific to a DAM.

2.2.6.1 PidTagMessageClass Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) MUST be set to "IPC.Microsoft Exchange 4.0.Deferred Action".

2.2.6.2 PidTagDamBackPatched Property

Type: **PtypBoolean** ([MS-OXCDATA] section 2.11.1)

The **PidTagDamBackPatched** property ([MS-OXPROPS] section 2.728) MUST be set to "FALSE" when the DAM is generated; it MUST be set to "TRUE" if the DAM was updated by the server as a result of a **RopUpdateDeferredActionMessages** request ([MS-OXCROPS] section 2.2.11.3).

2.2.6.3 PidTagDamOriginalEntryId Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

This **PidTagDamOriginalEntryId** property ([MS-OXPROPS] section 2.729) MUST be set to the EntryID of the delivered (target) message that the client has to process.

2.2.6.4 PidTagRuleProvider Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleProvider** property ([MS-OXPROPS] section 2.1024) MUST be set to the same value as the **PidTagRuleProvider** property on the rule or rules that have generated the DAM.

2.2.6.5 PidTagRuleFolderEntryId Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleFolderEntryId** property ([MS-OXPROPS] section 2.1012) MUST be set to the EntryID of the folder where the rule (4) that triggered the generation of this DAM is stored.

2.2.6.6 PidTagClientActions Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagClientActions** property ([MS-OXPROPS] section 2.704) is a binary buffer specifying the actions (3) the client is required to take on the message. The buffer MUST be packed according to the rule (4) action buffer format specified in section 2.2.5. The server is required to set values in this property according to the relevant actions (3) as they were set by the client when the rule (4) was created or changed by using the **RopModifyRules** ROP ([MS-OXCROPS] section 2.2.11.1). Note that the server can combine actions (3) from different rules (4) into one DAM, in which case the rule (4) actions (3) will be concatenated in the DAM's **PidTagClientActions** property by using the proper action (3) syntax specified in section 2.2.5.

2.2.6.7 PidTagRuleIds Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleIds** property ([MS-OXPROPS] section 2.1014) is a buffer obtained by concatenating the **PidTagRuleId** (section 2.2.1.3.1.1) values (8 bytes each) from all the rules (4) that contributed actions (3) in the **PidTagClientActions** property (section 2.2.6.6). The length of this binary property MUST be a multiple of 8 bytes.

2.2.6.8 PidTagDeferredActionMessageOriginalEntryId Property

Type: **PtypServerId** ([MS-OXCDATA] section 2.11.1)

The **PidTagDeferredActionMessageOriginalEntryId** property ([MS-OXPROPS] section 2.731) contains the server EntryID for the DAM message on the server. This property is set by the server when the DAM is created.

2.2.7 DEM Syntax

A DEM SHOULD be created by the server when an error is encountered while executing a rule (4). This process is specified in section 3.2.5.1.3. Extended rules (4) are not used in DEMs.

In addition to properties required on any message, as specified in [MS-OXCMSG] section 2.2.1, the following properties are specific to a DEM.

2.2.7.1 PidTagDeferredActionMessageOriginalEntryId Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) MUST be set to "IPC.Microsoft Exchange 4.0.Deferred Error".

2.2.7.2 PidTagRuleError Property

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

The **PidTagRuleError** property ([MS-OXPROPS] section 2.1011) MUST be set to one of the following values, indicating the cause of the error encountered during the execution of the rule (4).

Value	Meaning
0x00000001	Generic error that doesn't fall into any of the other categories.
0x00000002	Error opening the rules folder.
0x00000003	Error delivering the message.

Value	Meaning
0x00000004	Error while parsing the rule format.
0x00000005	Error processing the rule (4).
0x00000006	Error moving or copying the message to the destination folder.
0x00000007	Permission error moving or copying the message to the destination folder.
0x00000008	Error creating the DAM.
0x00000009	Error sending as another user.
0x0000000A	Error retrieving the reply template.
0x0000000B	Generic error while executing the rule (4) on the server.
0x0000000C	Error processing rule (4) due to mailbox quotas.
0x0000000D	Error processing the message due to the large number of recipients (2).
0x0000000E	Error copying or moving a message due to folder quotas.

2.2.7.3 PidTagRuleActionType Property

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

The **PidTagRuleActionType** property ([MS-OXPROPS] section 2.1009) MUST be set to the **ActionType** field value (see section 2.2.5.1.1) of the action (3) in the rule (4) that failed or set to 0x00000000 if the failure is not specific to an action (3). Related property: **PidTagRuleActionNumber** (section 2.2.7.4).

2.2.7.4 PidTagRuleActionNumber Property

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

The **PidTagRuleActionNumber** property ([MS-OXPROPS] section 2.1007) MUST be set to the zero-based index of the action (3) that failed or set to 0x00000000 if the failure is not specific to an action (3). (For example, if specific to an action (3), a property value of 0x00000000 means that the first action (3) failed, 0x00000001 means that the second action (3) failed.) The **ActionType** field value of the action (3) at this index MUST be the same value as the value of the **PidTagRuleActionType** property (section 2.2.7.3) in this DEM.

2.2.7.5 PidTagRuleProvider Property

Type: **PtypString** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleProvider** property (section 2.2.1.3.1.5) MUST be set to the same value as the **PidTagRuleProvider** property on the rule or rules that have caused the DEM to be generated.

2.2.7.6 PidTagDamOriginalEntryId Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagDamOriginalEntryId** property (section 2.2.6.3) MUST be set to the EntryID of the message that was being processed by the server when this error was encountered (that is, the "delivered message").

2.2.7.7 PidTagRuleFolderEntryId Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagRuleFolderEntryId** property (section 2.2.6.5) MUST be set to the EntryID of the folder where the rule (4) that triggered the generation of this DEM is stored.

2.2.7.8 PidTagRuleId Property

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

The **PidTagRuleId** (section 2.2.1.3.1.1) property MUST be set to the same value as the value of the **PidTagRuleId** property on the rule (4) that has generated this error.

2.2.8 Rules-Related Folder Properties

2.2.8.1 PidTagHasRules Property

Type: **PtypBoolean** ([MS-OXCDATA] section 2.11.1)

The **PidTagHasRules** property ([MS-OXPROPS] section 2.787) specifies whether rules (4) are set on a folder. This property MUST be set to "TRUE" if any rules (4) are set on a folder and "FALSE" otherwise. If this property does not exist, it is treated as though its value is "FALSE".

2.2.9 Rules-Related Message Properties

2.2.9.1 PidTagHasDeferredActionMessages Property

Type: **PtypBoolean** ([MS-OXCDATA] section 2.11.1)

The **PidTagHasDeferredActionMessages** property ([MS-OXPROPS] section 2.785) specifies whether a message has at least one associated DAM. This property MUST be set to "TRUE" if it does and "FALSE" otherwise. If this property does not exist, it is treated as though its value is "FALSE".

2.2.9.2 PidTagReplyTemplateId Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)

The **PidTagReplyTemplateId** property ([MS-OXPROPS] section 2.983) specifies the GUID for the reply template.

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.

The following abstract data model (ADM) object types are defined in this section:

Deferred Action Contents Table

3.1.1.1 Per Deferred Actions Contents Table

The deferred action **contents table** is represented by the **DeferredActionContentsTable** ADM object type. The client maintains a contents table that describes the DAMs and DEMs contained in the DAF. The client ensures that the rows in this table representing DAMs and DEMs are processed in a timely manner as specified in section 3.1.5.1.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Retrieving Existing Rules

When a higher layer needs to inspect the standard rules or needs to display these rules (4) to the user, the client MUST retrieve the rules (4) from the server using the **RopGetRulesTable ROP request** ([MS-OXCROPS] section 2.2.11.2) as specified in section 2.2.2. The higher level MUST use the returned table handle, as described in [MS-OXCTABL] section 1.5, to access rule (4) properties.

The table returned by the **RopGetRulesTable** ROP contains one rule (4) per row. The columns available in this table are the properties specified in section 2.2.1.3.1, and their values are the same as those the client set previously using a **RopModifyRules** ROP request ([MS-OXCROPS] section 2.2.11.1). If there isn't a value stored on the server for one of the rule (4) property columns, then when the client retrieves the rule (4) via a **RopGetRulesTable** ROP request, the server returns either a default value or an error for that column; which default values or errors are determined by the server implementation.

When a higher layer needs to inspect the extended rules or needs to display the extended rules to the user, the client MUST retrieve the FAI contents table of the folder of interest and use a **PropertyRestriction** restriction to restrict the folder to messages where the value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) is equal to "IPM.ExtendedRule.Message". For more details about retrieving an FAI contents table and restricting a table, see [MS-OXCFOLD] section 3.1.4.10 and [MS-OXCTABL] section 2.2.2.4.

3.1.4.2 Adding, Modifying, or Deleting Rules

This section describes the process of adding, modifying or deleting rules (4).

3.1.4.2.1 Adding, Modifying or Deleting Standard Rules

When the client modifies standard rules as a result of user interaction, it MUST do so using a **RopModifyRules** ROP request ([MS-OXCROPS] section 2.2.11.1), as specified in section 2.2.1.<9>

When adding a standard rule, the client MUST NOT set a value for the **PidTagRuleId** property (section 2.2.7.8) and MUST set values for the **PidTagRuleProvider** (section 2.2.7.5), **PidTagRuleCondition** (section 2.2.1.3.1.9), and **PidTagRuleActions** (section 2.2.1.3.1.10) properties on each rule (4) in the ROP request buffer. The client MAY set values for the **PidTagRuleUserFlags** (section 2.2.1.3.1.7) and **PidTagRuleProviderData** (section 2.2.1.3.1.8) properties for storing additional data. The client SHOULD send values for the other properties specified in section 2.2.1.3.1 in the ROP request buffer.

When modifying a standard rule, the client MUST send values for the **PidTagRuleId** property and MUST send values for properties that are to be changed, as specified in section 2.2.1.3.1.

When deleting a standard rule, the client MUST only send the value of the **PidTagRuleId** property in the ROP request buffer.

3.1.4.2.2 Adding, Modifying or Deleting Extended Rules

To add, modify, or delete an extended rule, a client adds, modifies, or deletes the FAI message representing that rule (4) respectively. The client uses standard message operations, as specified in [MS-OXCMSG] section 3.1.4.

When adding an extended rule, the client MUST set values for the **PidTagRuleMessageName** (section 2.2.4.1.1), **PidTagRuleMessageProvider** (section 2.2.4.1.7), **PidTagExtendedRuleMessageCondition**, (section 2.2.4.1.10), and **PidTagExtendedRuleMessageActions** (section 2.2.4.1.9) properties for each rule (4) on the FAI message representing that rule (4). The client MAY set values for the **PidTagRuleMessageUserFlags** (section 2.2.4.1.5) and **PidTagRuleMessageProviderData** ([MS-OXPROPS] section 2.1019) properties for storing additional data. The client SHOULD set values for the other properties on the FAI message, as specified in section 2.2.4.1.

When modifying an extended rule, the client MUST send values for properties that are to be changed, as specified in section 2.2.4.1.

When deleting an extended rule, the client MUST delete the FAI message representing that rule (4).

3.1.4.2.3 Creating Rules for Public Folders

When creating rules for public folders, the client MUST limit the conditions and actions (3) that are available for public folders to server-side rules by only using rule (4) actions (3) that can be executed by the server.

3.1.4.2.4 Creating Rich Client-Side Rules

To implement richer rules (4) functionality than provided by the server (for example, rules (4) that are evaluated when sending a message, the client can store additional rules (4) metadata that is opaque to the server. If the client does have metadata associated with rules (4) in the rules table, the client MUST store this metadata in a **Rule FAI message** stored in the **Inbox folder**. For more details about working with FAI messages, see [MS-OXCFOLD] and [MS-OXCMSG].

The Rule FAI message is an FAI message, as specified in [MS-OXCMSG]. The client MUST create (or open, if already present) the Rule FAI message in the Inbox folder. This message MUST be identified by the values of its **PidTagSubject** ([MS-OXCMSG] section 2.2.1.46) and **PidTagMessageClass** ([MS-OXCMSG] section 2.2.1.3) properties as follows: the value of the **PidTagMessageClass** property MUST be set to "IPM.RuleOrganizer"; the value of the **PidTagSubject** property MUST be set to "Outlook Rules Organizer".

Other properties on the Rule FAI message are up to the client application and MUST be treated by the server as opaque. Some clients<10> use the **PidTagRwRulesStream** property ([MS-OXPROPS] section 2.1029) on the Rule FAI message to store additional rule data that is opaque to the server. Other clients can use other opaque properties on the Rule FAI message for storing client-specific rules data.

3.1.4.2.5 Creating a Reply Template

Before creating a rule (4) that has an "OP_REPLY" or "OP_OOF_REPLY" value for the **ActionType** field, the client MUST first create a Reply template FAI message in the same folder as the rule (4).

The following steps specify how to create a Reply template:

1. Create a new FAI message in the folder.
2. Set the value of the **PidTagMessageClass** property ([MS-OXPROPS] section 2.855) to a string that has the prefix "IPM.Note.rules.ReplyTemplate." (for "OP_REPLY" values) or "IPM.Note.rules.OOFTemplate." (for "OP_OOF_REPLY" values).
3. Set the value of the **PidTagReplyTemplateId** property (section 2.2.9.2) with a newly generated GUID.
4. Set the value of **PidTagSubject** property ([MS-OXCMSG] section 2.2.1.46), the text of the message, and other message properties as desired.
5. Save the newly created message.
6. Get the value of the MID ([MS-OXCDATA] section 2.2.1.2) and FID ([MS-OXCDATA] section 2.2.1.1) from the saved message.

The value of the **PidTagReplyTemplateId** property generated by the client at step 3 is the value used by the **ReplyTemplateGUID** field in the Reply **ActionData** buffer specified in section 2.2.5.1.3.2.

For more details about creating and working with FAI messages, see [MS-OXCFOLD] and [MS-OXCMSG].

3.1.4.3 Downloading a Message to a Different Store

To download or move a message from the server to a different store, the client performs the following steps:

1. Retrieves the properties on the message.
2. Creates a new message with these properties.
3. Saves the message on a different store.
4. Deletes the message on the original store. (As a result, the EntryID that uniquely identifies this message in the messaging system can change.)

If the client changes the EntryID of a message that has the **PidTagHasDeferredActionMessages** property (section 2.2.9.1) set to TRUE, the client MUST send a **RopUpdateDeferredActionMessages** ROP ([MS-OXCROPS] section 2.2.11.3) to the server as specified in section 2.2.3, informing the server of the EntryID change, as soon as the EntryID of the DAM has been updated on the client.

3.1.5 Message Processing Events and Sequencing Rules

The messages specified in section 2.2 of this protocol are all sent by the client. The client processes the ROP response buffer associated with each message it sends as specified in section 2.2.1.2, section 2.2.2.2, and section 2.2.3.2. For more details on processing ROPs associated with rules (4), see [MS-OXCROPS] section 2.2.11.

3.1.5.1 Processing DAMs and DEMs

If the client creates any rules (4), the client SHOULD check the DAF for DAMs and DEMs placed in that folder and process the ones identified by the **PidTagRuleProvider** property value (section 2.2.1.3.1.5) the client supports. The DAF is a special folder that the server creates, as specified in section 3.2.1.3. The server places a message in the DAF either when it needs the client to perform an action (3) as a result of a client-side rule (DAM) or when it encounters a problem performing an action (3) of a server-side rule (DEM). When the server creates a DAM, it updates the **PidTagDeferredActionMessageOriginalEntryId** property (section 2.2.6.8), which is then used by the client in the **ServerEntryId** field of the **RopUpdateDeferredActionMessages** ROP request buffer (section 2.2.3).

After the client connects to the server, it inspects the contents of the DAF, as specified in [MS-OXCFOLD] section 3.2.5.14, for new DAMs or DEMs. The client processes DAMs and DEMs as specified in section 3.1.5.1.1 and section 3.1.5.1.2.

3.1.5.1.1 Processing a DAM

When processing a DAM, the client MUST first determine whether it has to process the DAM by inspecting the value of the **PidTagRuleProvider** property (section 2.2.7.5) on the DAM. If the value matches one of the rule provider strings the client supports, the client SHOULD process the DAM; otherwise, the client MUST ignore the DAM.

In addition to the **PidTagRuleProvider** property, when processing a DAM, the client can use any combination of the properties the server sets on the DAM as specified in section 2.2.6 to execute the rule (4). In particular, the client MUST use the value of the **PidTagDamOriginalEntryId** property (section 2.2.6.3) to identify the message it needs to take action (3) on, and it SHOULD use the value of the **PidTagClientActions** property (section 2.2.6.6) to identify what actions (3) it needs to execute on the message.

After processing a DAM, the client MUST delete the DAM. For more details about how to delete a message, see [MS-OXCFOLD] section 2.2.1.11.

3.1.5.1.2 Processing a DEM

When processing a DEM, the client MUST first determine whether it has to process the DEM by inspecting the value of the **PidTagRuleProvider** property (section 2.2.7.5) on the DEM. If the value matches one of the rule provider strings the client supports, the client SHOULD process the DEM at its earliest convenience; otherwise, the client MUST ignore the DEM.

In addition to the **PidTagRuleProvider** property, when processing a DEM, the client can use any combination of the properties the server sets on the DEM as specified in section 2.2.7. In particular,

the client SHOULD use the value of the **PidTagRuleError** property (section 2.2.7.2) to identify what error occurred, and it SHOULD use the values of the **PidTagRuleFolderEntryId** (section 2.2.7.7) and **PidTagRuleId** (section 2.2.7.8) properties if it needs to get more information from the rules table about the rule (4) that failed and return that information to the higher levels.

As a result of processing the DEM, the client SHOULD display an error to the user or take programmatic action (3) as a result of a rule (4) in error.

After processing a DEM, the client MUST delete the DEM. For more details about how to delete a message, see [MS-OXCFOLD] section 2.2.1.11.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

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

The following abstract data model (ADM) object types are defined in this section:

Mailbox

Message

Rules Table

Rule

3.2.1.1 Per Mailbox

Mailboxes are represented by the **Mailbox** ADM object type. The following ADM objects are maintained for each **Mailbox** ADM object type:

Mailbox.Message: An abstract representation of an e-mail message.

3.2.1.2 Per Message

An e-mail message is represented by the **Message** ADM object type.

3.2.1.3 Per Rules Table

The rules table is represented by the **RulesTable** ADM object type. The following ADM object is maintained for each **RulesTable** ADM object type:

RulesTables.Rule: A collection of rules that can be applied to incoming **Message** ADM object types.

3.2.1.4 Per Rule

A **rule (4)** is represented by the **Rule** ADM object type. The following ADM objects and states are maintained for each **Rule** ADM object type:

Rule.Enabled: True if this rule (4) is enabled and will execute when the conditions of the rule are met; otherwise, false.

Rule.OutOfOffice: True if the rule is executed only when the user is OOF; otherwise, false.

Rule.RuleData: Specifies the conditions that trigger the rule (4).

3.2.2 Timers

None.

3.2.3 Initialization

Prior to any client connecting to a mailbox, the server MUST ensure that the DAF has been created for that mailbox as specified in [MS-OXOSFLD] section 3.1.4.1. If a DAF for a mailbox has not been created or has not been found, then client-side rules and DEMs will not be processed by the client. The DAF SHOULD support notifications on its contents table object, as specified in [MS-OXCNOTIF].

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Returning and Maintaining the Rules Table

When a user creates or modifies a rule using the **RopModifyRules** ROP request ([MS-OXCROPS] section 2.2.11.1), the server MUST store this and all previously created rules. The server MUST also respond to a **RopGetRulesTable** ROP request ([MS-OXCROPS] section 2.2.11.2) by returning these rules to the client in the form of a rules table.

The server MUST also parse the rules (4) set on each folder according to the syntax specified in section 2.2.1 and evaluate and execute these rules (4) when messages are delivered to that folder.

3.2.4.2 Entering and Exiting the Out of Office State

When the mailbox enters the Out of Office state as specified in [MS-OXWOOF] section 2.2.4.1, the server MUST start processing rules (4) marked with the **ST_ONLY_WHEN_OOF** flag in the **PidTagRuleState** property (section 2.2.1.3.1.3). The server MUST also keep a list for rules (4) that have the **ST_KEEP_OOF_HIST** flag in the **PidTagRuleState** property specified in section 3.2.1.2.

When the mailbox exits the Out of Office state, the server MUST stop processing rules (4) marked with the **ST_ONLY_WHEN_OOF** flag in the **PidTagRuleState** property and clear the list for all rules (4). The semantics for processing rules (4) marked with the **ST_ONLY_WHEN_OOF** flag are specified in section 3.2.1.2.

3.2.4.3 Changing Rules Using a Server-Side Interface

The server can implement a user interface (for example, a server implementation of a client) that allows the user to modify all or some rules (4). Since the functionality provided by the standard client can exceed the functionality provided by the server as explained in section 3.1.4.2.4, if the

server modifies any rules (4) as a result of user interaction, the server MUST also delete the client-specific Rule FAI message specified in section 3.1.4.2.4. The server SHOULD warn the user making the change that doing so might lead to loss of specific rule (4) functionality implemented by the standard client.

3.2.5 Message Processing Events and Sequencing Rules

The following events are processed by a messaging server implementing this protocol. Note there is no particular sequence required for the ROP processing, other than that the server MUST send back a matching response for each ROP request sent by the client, as specified in [MS-OXCROPS].

3.2.5.1 Processing Incoming Messages to a Folder

When a message is either delivered to a private mailbox folder or posted to a public folder, the messaging server SHOULD evaluate the rules (4) that apply to the folder where the message was delivered. If a rule (4) moves the message to a folder where a different set of rules (4) exist, the server applies rules (4) recursively on the incoming message before executing any subsequent rules in the original folder.

A server can restrict the number of extended rules it executes on a folder.<11>

For each message delivered to a folder, the server evaluates each rule (4) in that folder in increasing order of the value of the **PidTagRuleSequence** property (section 2.2.1.3.1.2) in each rule (4). If two or more rules (4) have the same value for the **PidTagRuleSequence** property, the order in which the server evaluates these rules (4) is not defined.

The server MUST only evaluate rules (4) that are enabled; that is, rules (4) that have the **ST_ENABLED** flag set in the **PidTagRuleState** property (section 2.2.1.3.1.3).

The server MUST evaluate rules (4) that have the **ST_ONLY_WHEN_OOF** flag set in the **PidTagRuleState** property only when the mailbox is in an OOF state as specified in [MS-OXWOOF] section 2.2.4.1.

When executing a rule (4) whose condition evaluates to "TRUE" as per the restriction (2) in the **PidTagRuleCondition** property (section 2.2.1.3.1.9), then the server MUST either perform the actions (3) specified in the **PidTagRuleActions** property (section 2.2.1.3.1.10) associated with that rule (4) (in the case of a server-side rule) or generate a DAM for the client to process as specified in section 3.2.5.1.2. Following is a description of what the server does when it executes each action (3) type, as specified in section 2.2.5.1.1, for an incoming message:

- "OP_MOVE": The server MUST place a copy of the message in the folder specified in the action buffer structure and delete the original message; if multiple "OP_MOVE" operations apply to the same message, the server SHOULD create multiple copies of the message and then delete the original message.
- "OP_COPY": The server MUST place a copy of the message in the folder specified in the action buffer structure.
- "OP_REPLY": The server MUST use properties from the reply template (for example, body text properties, recipients (2) on the template) and from the original message (for example, the sender of the message) to create a reply to the message and then send the reply. The server MUST NOT send a reply if the **PidTagAutoResponseSuppress** property ([MS-OXOMSG] section 2.2.1.73) on the message that has the 0x00000020 bit set. For more details on suppression of automatic replies, see [MS-OXCMAIL] section 2.2.3.2.14. The server SHOULD also avoid sending replies to automatically generated messages, which are identified by the

PidTagAutoForwarded property ([MS-OXCMSG] section 2.2.1.20), to avoid generating endless autoreply loops.

- "OP_OOF_REPLY": The server MUST behave as specified for the "OP_REPLY" action (3). In addition, the server SHOULD set the value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) on the reply message to "IPM.Note.rules.OOFTemplate". This message class value is a prefix, and the client can append a client-specific value at the end; for example, a client can instead request that the server set the value of the **PidTagMessageClass** property in this circumstance to "IPM.Note.rules.OOFTemplate.Microsoft".<12> The server MUST NOT send a reply if the **PidTagAutoResponseSuppress** property on the message has the 0x00000010 bit set. For more information on suppression of automatic replies, see [MS-OXCMAIL] section 2.2.3.2.14.
- "OP_DEFER_ACTION": The server MUST generate a DAM as specified in section 3.2.5.1.2. The server MUST also set the **PidTagHasDeferredActionMessages** property (section 2.2.9.1) to "TRUE" on the message.
- "OP_FORWARD": The server MUST forward the message to the recipients (2) specified in the action buffer structure. The server SHOULD NOT<13> forward messages that were forwarded to the sender.
- "OP_DELEGATE": the server MUST resend the message to the recipients (2) specified in the action buffer structure. The server also MUST set the values of the following properties to match the current user's properties in the **address book**:
  - The **PidTagReceivedRepresentingEntryId** property ([MS-OXOMSG] section 2.2.1.25) MUST be set to the same value as the mailbox user's **PidTagEntryId** property ([MS-OXOABK] section 2.2.3.3).
  - The **PidTagReceivedRepresentingAddressType** property ([MS-OXOMSG] section 2.2.1.23) MUST be set to the same value as the mailbox user's **PidTagAddressType** property ([MS-OXOABK] section 2.2.3.13).
  - The **PidTagReceivedRepresentingEmailAddress** property ([MS-OXOMSG] section 2.2.1.24) MUST be set to the same value as the mailbox user's **PidTagEmailAddress** property ([MS-OXOABK] section 2.2.3.14).
  - The **PidTagReceivedRepresentingName** property ([MS-OXOMSG] section 2.2.1.26) MUST be set to the same value as the mailbox user's **PidTagDisplayName** property ([MS-OXCFOLD] section 2.2.2.2.2.4).
  - The **PidTagReceivedRepresentingSearchKey** property ([MS-OXOMSG] section 2.2.1.27) MUST be set to the same value as the mailbox user's **PidTagSearchKey** property ([MS-OXCPRPT] section 2.2.1.9).
  - The **PidTagDelegatedByRule** property ([MS-OXOMSG] section 2.2.1.80) MUST be set to "TRUE".
- "OP_BOUNCE": The server MUST send a reply message to the sender detailing why the sender's message couldn't be delivered to the user's mailbox; the original message MUST NOT appear in the user's mailbox.
- "OP_TAG": The server MUST set on the message the property specified in the action buffer structure.
- "OP_DELETE": The server MUST delete the message. The server MUST stop evaluating subsequent rules (4) on the message except for Out of Office rules.

- "OP_MARK_AS_READ": the server MUST set the **MSGFLAG_READ** flag (0x00000001) in the **PidTagMessageFlags** property ([MS-OXPROPS] section 2.859) on the message.

If the server fails to execute a rule (4) action (3), the server MUST generate a DEM as specified in section 3.2.5.1.3.

The server MUST place all DAMs and DEMs that it creates as a result of running any rule (4) in any folder into the DAF.

3.2.5.1.1 Processing Out of Office Rules

The server evaluates and executes Out of Office rules only when the mailbox is in an Out of Office state, as specified in [MS-OXWOOF] section 2.2.4.1.

If a rule (4) has the **ST_KEEP_OOF_HIST** flag set in the **PidTagRuleState** property (section 2.2.1.3.1.3), the server MUST keep a history of recipients for that rule (4) and check whether the sender of the delivered message appears in the list for that rule (4). If the sender is on the list, the server MUST NOT evaluate the rule (4). If not and the rule (4) condition evaluates to "TRUE", the server MUST add the sender to the list of recipients (2) for the rule (4) in addition to executing the rule (4) action (3). If the rule (4) condition evaluates to "FALSE", no additional action (3) needs to be taken.

3.2.5.1.1.1 Interaction Between ST_ONLY_WHEN_OOF and ST_EXIT_LEVEL Flags

When the Out of Office state is set on the mailbox, as specified in [MS-OXWOOF], and a rule (4) condition evaluates to "TRUE", if the rule (4) has the **ST_EXIT_LEVEL** flag specified in section 2.2.1.3.1.3 set, then the server MUST NOT evaluate subsequent rules (4) that do not have the **ST_ONLY_WHEN_OOF** flag set. Subsequent rules (4) that have the **ST_ONLY_WHEN_OOF** flag set MUST be evaluated.

3.2.5.1.2 Generating a DAM

A server MUST generate a DAM when a rule (4) condition evaluates to "TRUE" but the server cannot perform the actions (3) specified in the rule (4). When the server generates DAMs for a message, the server MUST set the value of the **PidTagHasDeferredActionMessages** property (section 2.2.9.1) on the message to "TRUE".

The server MUST generate the DAM in the following manner:

- Create a new message (DAM) in the DAF.
- Set the property values on the DAM as specified in section 2.2.6.
- Save the DAM.

The server can pack information about more than one "OP_DEFER_ACTION" actions (3), as specified in section 2.2.5.1.1, for any given message into one DAM. The server SHOULD do this when there are more than one "OP_DEFER_ACTION" actions (3) that belong to the same rule provider. The server MUST generate separate DAMs for "OP_DEFER_ACTION" actions (3) that belong to separate rule providers.

3.2.5.1.3 Handling Errors During Rule Processing (Creating a DEM)

A server SHOULD generate a DEM when it encounters an error processing a rule (4) on an incoming message. The server SHOULD also generate a DEM if it fails to create a DAM for a specific rule (4).

The server MUST generate the DEM in the following manner:

- Create a new message (DEM) in the DAF.
- Set the property values on the DEM as specified in section 2.2.7.
- Save the DEM.

The first time the server finds a server-side rule to be in error and has generated a DEM for it, the server SHOULD set the **ST_ERROR** flag in the **PidTagRuleState** property (section 2.2.1.3.1.3) of that rule (4). Examination of the **ST_ERROR** flag on subsequent operations is used to prevent creating multiple DEMs with the same error information.

3.2.5.2 Receiving a RopModifyRules ROP Request

When receiving a **RopModifyRules** ROP request ([MS-OXCROPS] section 2.2.11.1), the server MUST parse the request according to the syntax specified in section 2.2.1. If the server encounters an error while parsing the request buffer, or if any data in the request buffer is incorrect, the server MUST return an error in the **ReturnValue** field in the response buffer.

If the server successfully parses the data in the request buffer and is able to process all requests for adding, modifying, and deleting rules (4) present in the request buffer, the server MUST return 0x00000000 as the value of the **ReturnValue** field in the response buffer. The server MUST assign a value for the **PidTagRuleId** property (section 2.2.7.8) for each rule (4) that has been added by the **RopModifyRules** ROP request. The value of the **PidTagRuleId** property on each rule (4) MUST be unique in that folder.

The server can limit the rules (4) it allows on a folder to a certain number of rules (4) or to a total aggregate size of rules.<14> If a **RopModifyRules** request causes the rules (4) to exceed the limit, the server MUST return the ecNotEnoughMemory (0x8007000E) error in the **ReturnValue** field of the **RopModifyRules** response. Regardless of the limit, the server SHOULD<15> save all changes specified by the **RopModifyRules** request.

The server MUST update the value of the **PidTagHasRules** property (section 2.2.8.1) when rules (4) change on a folder. The value of this property MUST be set to "TRUE" if any rules (4) are set in that folder and to "FALSE" otherwise. The server SHOULD start using the newly modified rules (4) when processing messages delivered to that folder as soon as it successfully processes the **RopModifyRules** ROP request. Any rules that exceed the limit are disabled during message delivery to the folder.

The following error codes can be returned by this ROP.

Error code name	Value	Meaning
ecInvalidParam	0x80070057	One or more of the **x** bits on the **TableFlags** field is set to a nonzero value.
ecNotEnoughMemory	0x8007000E	The number of rules (4) has been exceeded or the aggregate size of rules (4) has been exceeded.

3.2.5.3 Receiving a RopGetRulesTable ROP Request

When receiving a **RopGetRulesTable** ROP request ([MS-OXCROPS] section 2.2.11.2), the server MUST parse the request according to the syntax specified in section 2.2.2. If the server encounters an error parsing the request buffer, or if any data in the request buffer is incorrect, the server MUST

return an error in the **ReturnValue** field of the ROP response buffer. A list of common error return values are described in [MS-OXCDATA] section 2.4.

If the server successfully parses the data in the ROP request buffer, it MUST return 0x00000000 as the value of the **ReturnValue** field in the response buffer and MUST return a valid table handle through which the client can access the folder rules (4) using table specific ROPs defined in [MS-OXCTABL].

The following error code can be returned by this ROP.

Error code name	Value	Meaning
ecNotSupported	0x80040102	One or more of the **x** bits on the **TableFlags** field is set to a nonzero value.<16>

3.2.5.4 Receiving a RopUpdateDeferredActionMessages ROP Request

When receiving a **RopUpdateDeferredActionMessages** ROP request ([MS-OXCROPS] section 2.2.11.3), the server MUST parse the request according to the syntax specified in section 2.2.3. If the server encounters an error parsing the ROP request buffer, or if any data in the request buffer is incorrect, the server MUST return an error in the **ReturnValue** field of the ROP response buffer. For a list of common error return values, see [MS-OXCDATA] section 2.4.

If the server successfully parses the data in the ROP request buffer, it MUST return 0x00000000 as the value of the **ReturnValue** field in the response buffer. The server also MUST find all DAMs that have the value of the **PidTagDeferredActionMessageOriginalEntryId** property (section 2.2.6.8) equal to the value in the **ServerEntryId** field of the **RopUpdateDeferredActionMessages** ROP request buffer, as specified in section 2.2.3. The server MUST then change the value of the **PidTagDeferredActionMessageOriginalEntryId** property on each DAM it finds to the value passed in the **ClientEntryId** field of the same ROP request buffer. The server MUST also set the value of the **PidTagDamBackPatched** property (section 2.2.6.2) to "TRUE" on any DAM that it changed.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

Starting with a "clean" folder (that is, a folder with no rules (4)), here is a sample sequence of ROP request buffers and ROP response buffers that a client and a server might exchange. Note that the examples listed here only show the relevant portions of the specified ROPs; this is not the final byte sequence that gets transmitted over the wire. Also note that the data for a multibyte field appear in little-endian format, with the bytes in the field presented from least significant to most significant. Generally speaking, these ROP request buffers are packed with other ROP request buffers, compressed and packed in one or more remote procedure calls (RPCs) as described in [MS-OXCROPS]. These examples assume the client has already successfully logged on to the server and opened the folder on which it will modify the rules (4).

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

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

The value at the far left is the offset of the following bytes into the buffer, expressed in hexadecimal notation. Following the offset is a series of up to 16 bytes, with each two-character sequence describing the value of one byte in hexadecimal notation. Here, for example, the byte "53" (01010011) is located 0x82 bytes (130 bytes) from the beginning of the buffer. The dash between the eighth byte ("44") and the ninth byte ("4C") has no semantic value and serves only to distinguish the 8-byte boundary for readability purposes.

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

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

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

PropertyTag: 0x66760003 (**PidTagRuleSequence** (section 2.2.1.3.1.2))

PropertyValue: 10

Generally speaking, interpreted values will be shown in their native format, interpreted from the raw byte sequence as described in the specific section. Here, the byte sequence "0a 00 00 00" has been interpreted as a **ULONG** ([MS-DTYP]) with a value of 10 although the type of the **PidTagRuleSequence** property is **PtypInteger32** ([MS-OXCDATA] section 2.11.1.5).

4.1 Adding a New Rule

In this example, a user wants to add a rule (4) to move e-mail messages to a folder named "X" when the subject contains the phrase "Project X". The client sends a **RopModifyRules** ROP request ([MS-OXCROPS] section 2.2.11.1), in the buffer format specified in section 2.2.1.

4.1.1 Client Request Buffer

A complete ROP request buffer in this example would appear as follows.

```
0000: 41 00 01 00 01 00 01 08-00 1f 00 82 66 50 00 72
0010: 00 6f 00 6a 00 65 00 63-00 74 00 20 00 58 00 00
0020: 00 03 00 76 66 0a 00 00-00 03 00 77 66 01 00 00
0030: 00 fd 00 79 66 03 01 00-01 00 1f 00 37 00 1f 00
0040: 37 00 50 00 72 00 6f 00-6a 00 65 00 63 00 74 00
0050: 20 00 58 00 00 00 fe 00-80 66 01 00 d0 00 01 00
0060: 00 00 00 00 00 00 00 01-ad 00 00 00 00 00 38 a1
0070: bb 10 05 e5 10 1a a1 bb-08 00 2b 2a 56 c2 00 00
0080: 45 4d 53 4d 44 42 2e 44-4c 4c 00 00 00 00 00 00
0090: 00 00 1b 55 fa 20 aa 66-11 cb 9b c8 00 aa 00 2f
00a0: c4 5a 0c 00 00 00 4f 4c-45 58 44 4f 47 31 32 00
00b0: 2f 6f 3d 46 69 72 73 74-4f 72 67 61 6e 69 7a 61
00c0: 74 69 6f 6e 2f 6f 75 3d-45 78 63 68 61 6e 67 65
00d0: 20 41 64 6d 69 6e 69 73-74 72 61 74 69 76 65 20
00e0: 47 72 6f 75 70 20 28 46-59 44 49 42 4f 48 46 32
00f0: 33 53 50 44 4c 54 29 2f-63 6e 3d 52 65 63 69 70
0100: 69 65 6e 74 73 2f 63 6e-3d 74 65 72 72 79 6d 61
0110: 68 44 31 32 2d 31 00 15-00 01 04 00 00 00 01 72
0120: 00 0c 00 00 00 00 00 00-00 00 00 00 00 00 1f 00
0130: 81 66 52 00 75 00 6c 00-65 00 4f 00 72 00 67 00
0140: 61 00 6e 00 69 00 7a 00-65 00 72 00 00 00 03 00
0150: 83 66 00 00 00 00 02 01-84 66 10 00 01 00 00 00
0160: 01 00 00 00 55 55 55 55-d1 44 e3 40
```

The first 6 bytes refer to the **RopId**, **LogonId**, **InputHandleIndex**, **ModifyRulesFlags**, and **RulesCount** fields of the **RopModifyRules** format as described in [MS-OXCROPS] section 2.2.11.1.

```
0000: 41 00 01 00 01 00
```

RopId: 0x41 (**RopModifyRules**)

LogonId: 0x00

InputHandleIndex: 0x01

ModifyRulesFlags: 0x00

RulesCount: 0x0001

The first and only **RuleData** structure for this request begins at byte 0x0006. The next 3 bytes are the **RuleDataFlags** and **PropertyValueCount** fields:

```
0006: 01 08 00
```

RuleDataFlags: 0x01 (ROW_ADD)

PropertyValueCount: 0x0008

The first of the eight **TaggedPropertyValues** fields begin at byte 0x0009. They are summarized below. For more information on the **PropertyValue** structure format, see [MS-OXCDATA] section 2.11.2.

```
0009: 1f 00 82 66 50 00 72 00-6f 00 6a 00 65 00 63-00
0019: 74 00 20 00 58 00 00 00
```

PropertyTag: 0x6682001F (**PidTagRuleName** (section 2.2.1.3.1.4))

PropertyValue: Unicode string: "Project X"

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

PropertyTag: 0x66760003 (**PidTagRuleSequence** (section 2.2.1.3.1.2))

PropertyValue: 0x0000000A

```
0029: 03 00 77 66 01 00 00-00
```

PropertyTag: 0x66770003 (**PidTagRuleState** (section 2.2.1.3.1.3))

PropertyValue: 0x00000001 (ST_ENABLED)

```
0031: fd 00 79 66 03 01 00 01-00 1f 00 37 00 1f 00 37
0041: 00 50 00 72 00 6f 00 6a-00 65 00 63 00 74 00 20
0051: 00 58 00 00 00
```

PropertyTag: 0x667900FD (**PidTagRuleCondition** (section 2.2.1.3.1.9))

PropertyValue: "RES_CONTENT" condition, **FuzzyLevel** of 0x00010001 (FL_SUBSTRING | FL_IGNORECASE), where **PropertyTag** 0x0037001F (**PidTagSubject** ([MS-OXPROPS] section 2.1096) contains "Project X". For more information, see section 2.2.1.

```
0056: fe 00 80 66 01 00 d0 00-01 00 00 00 00 00 00 00
0066: 00 01 ad 00 00 00 00 00-38 a1 bb 10 05 e5 10 1a
0076: a1 bb 08 00 2b 2a 56 c2-00 00 45 4d 53 4d 44 42
0086: 2e 44 4c 4c 00 00 00 00-00 00 00 00 1b 55 fa 20
0096: aa 66 11 cb 9b c8 00 aa-00 2f c4 5a 0c 00 00 00
00a6: 4f 4c 45 58 44 4f 47 31-32 00 2f 6f 3d 46 69 72
00b6: 73 74 4f 72 67 61 6e 69-7a 61 74 69 6f 6e 2f 6f
00c6: 75 3d 45 78 63 68 61 6e-67 65 20 41 64 6d 69 6e
00d6: 69 73 74 72 61 74 69 76-65 20 47 72 6f 75 70 20
00e6: 28 46 59 44 49 42 4f 48-46 32 33 53 50 44 4c 54
00f6: 29 2f 63 6e 3d 52 65 63-69 70 69 65 6e 74 73 2f
0106: 63 6e 3d 74 65 72 72 79-6d 61 68 44 31 32 2d 31
0116: 00 15 00 01 04 00 00 00-01 72 00 0c 00 00 00 00
0126: 00 00 00 00 00 00 00 00
```

PropertyTag: 0x668000FE (**PidTagRuleActions** (section 2.2.1.3.1.10))

PropertyValue: 0x0001 actions (3), 0x00D0 bytes long, to **ActionType** is 0x01 ("OP_MOVE"), **ActionFlavor** is 0x00000000, **ActionFlags** is 0x00000000, **FolderInThisStore** is 0x01, followed

by a **StoreEID** 0xAD bytes long, followed by a **FolderEID** 0x15 bytes long. For more details, see section 2.2.5.

```
012e: 1f 00 81 66 52 00 75 00-6c 00 65 00 4f 00 72 00
013e: 67 00 61 00 6e 00 69 00-7a 00 65 00 72 00 00 00
```

PropertyTag: 0x6681001F (**PidTagRuleProvider** (section 2.2.1.3.1.5))

PropertyValue: Unicode string: "RuleOrganizer"

```
014e: 03 00 83 66 00 00 00 00
```

PropertyTag: 0x66830003 (**PidTagRuleLevel** (section 2.2.1.3.1.6))

PropertyValue: 0x00000000

```
0156: 02 01 84 66 10 00 01 00-00 00 01 00 00 00 55 55
0166: 55 55 d1 44 e3 40
```

PropertyTag: 0x66840102 (**PidTagRuleProviderData** (section 2.2.1.3.1.8)

PropertyValue: BLOB, 0x0010 bytes long, set by the client.

4.1.2 Server Responds to Client Request

A complete ROP response buffer in this example would appear as follows.

```
0000: 41 01 00 00 00 00
```

RopId: 0x41 (**RopModifyRules** ([MS-OXCROPS] section 2.2.11.1))

InputHandleIndex: 0x01

ReturnValue: 0x00000000. This response indicates the client has successfully created the rule (4).

4.2 Displaying Rules to the User

In this example, a client is required to display a list of active rules (4) on a folder to a user. The client sends a **RopGetRulesTable** ROP request ([MS-OXCROPS] section 2.2.11.2), using the buffer format specified in section 2.2.2. The client also sends **RopSetColumns** ([MS-OXCROPS] section 2.2.5.1) and **RopQueryRows** ROP requests ([MS-OXCROPS] section 2.2.5.4), using the buffer format described in [MS-OXCROPS] and [MS-OXCTABL].

4.2.1 Client Request for a Rules Table

A complete ROP request buffer to request a rules table would appear as follows.

```
0000: 3f 00 00 01 40
```

RopId: 0x3F (**RopGetRulesTable** ([MS-OXCROPS] section 2.2.11.2))

LogonId: 0x00

InputHandleIndex: 0x00

OutputHandleIndex: 0x01

TableFlags: 0x40 (specifying a Unicode table)

The client can also simultaneously send other ROP request buffers (in the same **RPC**) to format the table or to get rows from it. These further requests can reference the **OutputHandleIndex** field (1 in this example) to specify the table to act on. For more information, see [MS-OXCROPS] and [MS-OXCDATA].

In this case, to format the table and read its rows, the client also sends a **RopSetColumns** ROP request ([MS-OXCROPS] section 2.2.5.1):

```
0000: 12 00 01 00 03 00 14 00-74 66 02 01 84 66 1f 00
0010: 82 66
```

RopId: 0x12 (**RopSetColumns**)

LogonId: 0x00

InputHandleIndex: 0x01

WantAsync: 0x00 (Wait)

PropertyTagCount: 3

PropertyTag1: 0x66740014 (**PidTagRuleId** ([MS-OXPROPS] section 2.1013))

PropertyTag2: 0x66840102 (**PidTagRuleProviderData** (section 2.2.1.3.1.8))

PropertyTag3: 0x6682001F (**PidTagRuleName** ([MS-OXPROPS] section 2.1023))

The client also sends a **RopQueryRows** ROP request ([MS-OXCROPS] section 2.2.5.4) to gather rows from the table.

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

RopId: 0x15 (**RopQueryRows**)

LogonId: 0

InputHandleIndex: 1

WantCurrentRow: "FALSE" (Advance)

WantForwardRead: "TRUE" (forward reading)

RowCount: 50

In this example, the handle array at the end of the RPC contains the following bytes.

```
0000: 23 02 00 00 ff ff ff ff
```

HandleIndex 0: 0x00000223

HandleIndex 1: 0xFFFFFFFF

Note that the **HandleIndex**[0] field is referenced only in the **RopGetRulesTable** ROP request – it refers to a table handle previously returned by the **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1) (the Inbox, for example).The **HandleIndex**[1] field is referenced by the **RopGetRulesTable** (as the new rules table index), the **RopSetColumns** (as the referenced table) and **RopQueryRows** (as the referenced table) ROP calls. The actual server handle does not yet exist, so the client fills in 0xFFFFFFFF temporarily.

4.2.2 Server Responds to Client Requests

The client has sent three separate ROP request buffers (**RopGetRulesTable** ([MS-OXCROPS] section 2.2.11.2), **RopSetColumns** ([MS-OXCROPS] section 2.2.5.1), and **RopQueryRows** ([MS-OXCROPS] section 2.2.5.4)), and the server responds with three ROP response buffers.

```
0000: 3f 01 00 00 00 00
```

RopId: 0x3F (**RopGetRulesTable**)

InputHandleIndex: 1

ReturnValue: 0x00000000. This response indicates the client has successfully gotten a handle to the rules table for the specified folder.

```
0000: 12 01 00 00 00 00 00
```

RopId: 0x12 (**RopSetColumns**)

InputHandleIndex: 1

ReturnValue: 0x00000000. This response indicates the client has successfully set the columns of the rules table.

CompletionStatus: 0x00 (TBLSTAT_COMPLETE ([MS-OXCTABL] section 2.2.2.1.3))

The response to the **RopQueryRows** ROP request buffer is slightly more verbose.

```
0000: 15 01 00 00 00 00 02 01-00 00 01 00 00 00 01 3f
0010: f8 56 10 00 01 00 00 00-01 00 00 00 55 55 55 55
0020: d1 44 e3 40 50 00 72 00-6f 00 6a 00 65 00 63 00
0030: 74 00 20 00 58 00 00 00
```

The first 9 bytes of a **RopQueryRows** **ROP response** contain data about the response.

```
0000: 15 01 00 00 00 00 02 01-00
```

RopId: 0x15 (**RopQueryRows**)

InputHandleIndex: 1

ReturnValue: 0x00000000. This response indicates the **RopQueryRows** ROP call was successful.

Bookmark: 0x02 (BOOKMARK_END ([MS-OXCTABL] section 2.2.2.1.1))

RowCount: 1

This is followed by the row property array beginning at byte 0x0009, which in this example contains one row (indicated by the **RowCount** field). It is not possible to interpret this response without the context of the earlier **RopSetColumns** ROP request because its format is based on the number of requested columns and the data type of each column.

```
0009: 00 01 00 00 00 01 3f f8-56 10 00 01 00 00 00 01
0019: 00 00 00 55 55 55 55 d1-44 e3 40 50 00 72 00 6f
0029: 00 6a 00 65 00 63 00 74-00 20 00 58 00 00 00
```

Has Flag: "FALSE"

Property 1: 0x56F83F0100000001

Property 2: 0x10 byte binary array

Property 3: "Project X"

Property 1, **Property 2**, and **Property 3** correspond to the **PidTagRuleId** (section 2.2.1.3.1.1), **PidTagRuleProviderData** (section 2.2.1.3.1.8), and **PidTagRuleName** (section 2.2.1.3.1.4) properties specified by the earlier **RopSetColumns** ROP request. For more information, see [MS-OXCROPS] and [MS-OXCTABL].

At the end of the three ROP response buffers, the handle table is as follows.

```
0000: 23 02 00 00 21 02 00 00
```

Handle 0: 0x00000223

Handle 1: 0x00000221

Note that the server has returned a proper handle for the rules table (0x00000221). The client uses this handle for any further requests relating to the rules table.

4.3 Deleting a Rule

In this example, a client is required to delete the rule (4) created in section 4.1 using the **RopModifyRules** ROP ([MS-OXCROPS] section 2.2.11.1). The client sends a **RopModifyRules** ROP request, using the buffer format described in section 2.2.1.

4.3.1 Client Request Buffer

A complete ROP request buffer in this example would appear as follows.

```
0000: 41 00 00 00 01 00 04 01-00 14 00 74 66 01 00 00
0010: 00 01 3f f8 56
```

The first 6 bytes refer to the **RopId**, **LogonId**, **InputHandleIndex**, **ModifyRulesFlags**, and **RulesCount** fields of the **RopModifyRules** format ([MS-OXCROPS] section 2.2.11.1) as described in section 2.2.1.

```
0000: 41 00 00 00 01 00
```

RopId: 0x41 (**RopModifyRules**)

LogonId: 0x00

InputHandleIndex: 0x00

ModifyRulesFlags: 0x00

RulesCount: 0x0001

The first and only **RuleData** structure for this request begins at byte 0x0006. The next 3 bytes are the **RuleDataFlags** and **PropertyValueCount** fields:

```
0006: 04 01 00
```

RuleDataFlags: 0x04 (ROW_REMOVE)

PropertyValueCount: x0001

The only **TaggedPropertyValue** begins at byte 0x0009. It is summarized below. For more information on property packing, see [MS-OXCDATA].

```
0009: 14 00 74 66 01 00 00 00-01 3f f8 56
```

PropertyTag: 0x66740014 (**PidTagRuleId** (section 2.2.7.8))

PropertyValue: 0x56F83F0100000001

4.3.2 Server Responds to Client Request

A complete ROP response buffer in this example appear as follows.

```
0000: 41 00 00 00 00 00
```

RopId: 0x41 (**RopModifyRules** ([MS-OXCROPS] section 2.2.11.1))

InputHandleIndex: 0x00

ReturnValue: 0x00000000. This response indicates the client has successfully removed the rule (4).

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to this protocol. General security considerations pertaining to the underlying RPC-based transport apply, as described in [MS-OXCROPS].

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

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

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

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

<1> Section 2.2.1.3.1.3: The Exchange 2007 implementation uses bit flags 0x00000080 and 0x00000100 to store information about Out of Office functionality; these bit flags are ignored by Office Outlook 2003 and Exchange 2003. Bit flag 0x00000080 is used to disable a specific Out of Office rule on Exchange 2007. Bit flag 0x00000100 has the same semantics as the **ST_ONLY_WHEN_OOF** bit flag on Exchange 2007. The rest of the flags are reserved for future use.

<2> Section 2.2.1.3.1.5: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 define the following well-known rule provider strings:

"MSFT:TDX Rules", which is used by public folder rules

"MSFT:TDX OOF rules", which is used by Out of Office rules in the Inbox folder

"RuleOrganizer" + user defined string, which is used for user-defined rules in the Inbox folder

"Schedule+ EMS Interface", which is used to assist with delegates

"JunkEmailRule", which is a rule that is created to help with Junk E-mail filtering

"MSFT:MR", which is a rule that assists the "Moderator" role on a public folder

"MSFT:Public.Folder.FormRestrictions", which is used by public folder rules

"ExchangeMailboxRules14", which is for rules that are specific to Exchange 2010 and Exchange 2013; rules with this provider string are not processed by Office Outlook 2003 or Office Outlook 2007.

<3> Section 2.2.2.1: Exchange 2007 ignores the **x** bits and does not return an error for nonzero values.

<4> Section 2.2.5.1: Office Outlook 2003, Office Outlook 2007, Outlook 2010, and Outlook 2013 set the **ActionFlags** field to 0x00000000.

<5> Section 2.2.5.1.2: Exchange 2003 and Exchange 2007 do not support forwarding messages as SMS text messages.

<6> Section 2.2.5.1.3.4.1: Exchange 2003 and Exchange 2007 set this value to 0x01. Exchange 2010 and Exchange 2013 set this value to 0x00.

<7> Section 2.2.5.1.3.4.1: Exchange 2003 and Exchange 2007 also require the **PidTagEntryId** property for action "OP_FORWARD".

<8> Section 2.2.5.1.3.7: Exchange 2003, Exchange 2007, Exchange 2010, and Exchange 2013 perform a **hard delete**, as described in [MS-OXCFOLD], but this is not required for the protocol.

<9> Section 3.1.4.2.1: Office Outlook 2003 and Office Outlook 2007 are only adding, modifying, and deleting rules on the following folders and ignore rules set on any other folder or folders.

- The Inbox folder, as described in [MS-OXOSFLD].
- Any public folder, as described in [MS-OXCSTOR], where the user has access permissions; extended rules are not set or evaluated on public folders.

<10> Section 3.1.4.2.4: Office Outlook 2003, Office Outlook 2007, Outlook 2010, and Outlook 2013 use the **PidTagRwRulesStream** property ([MS-OXPROPS] section 2.1029) to store opaque rules data.

<11> Section 3.2.5.1: Exchange 2003 by default will only process the first extended rule it encounters per folder. Other extended rules are ignored. Exchange 2007 by default will process the standard rule for a message but will only process the first two extended rules it encounters per folder. These settings are configurable by the administrator.

<12> Section 3.2.5.1: When Office Outlook 2007 creates a Reply template, it requests that the server set the prefix to "IPM.Note.rules.OofTemplate.Microsoft".

<13> Section 3.2.5.1: Exchange 2007 forwards messages that have been forwarded to the sender.

<14> Section 3.2.5.2: Exchange 2007, Exchange 2010, and Exchange 2013 limit the total aggregate size of standard rules (4) stored by a user to between 32 kilobytes (KB) and 256 KB; the exact limit is configured by the server administrator. Exchange 2007 and Exchange 2010 also enforce a limit of 500 disabled rules. Exchange 2013 does not enforce a limit on disabled rules. Exchange 2003 limits the total aggregate size of all rules to 32 KB.

<15> Section 3.2.5.2: Exchange 2003, Exchange 2007, and Exchange 2010 do not save any of the changes when the changes push the rules (4) beyond the limit.

<16> Section 3.2.5.3: Exchange 2007 ignores the **x** bits and returns ecSuccess in this case.

7 Change Tracking

This section identifies changes that were made to the [MS-OXORULE] protocol document between the July 2012 and October 2012 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

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

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

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

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

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

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

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

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2.2 Informative References	Added the reference [MS-OXPROTO].	N	Content updated.
1.4 Relationship to Other Protocols	Added informative reference information for overview of relationships between this and other protocols.	N	Content updated.
2.2.1.3.1.5 PidTagRuleProvider Property	Added Exchange 2013 to the product behavior note.	Y	Product behavior note updated.
2.2.5.1 Action Block Buffer Format	Added Outlook 2013 to the product behavior note.	Y	Product behavior note updated.
2.2.5.1.3.4.1 RecipientBlock Data Buffer Packet Structure	Added Exchange 2013 to the product behavior note.	Y	Product behavior note updated.
2.2.5.1.3.7 OP_DELETE or OP_MARK_AS_READ Data Buffer Format	Added Exchange 2013 to the product behavior note.	Y	Product behavior note updated.
3.1.4.2.4 Creating Rich Client-Side Rules	Added Outlook 2013 to the product behavior note.	Y	Product behavior note updated.
3.2.5.2 Receiving a RopModifyRules	Clarified the server's behavior when the rules limit is exceeded.	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
ROP Request			
3.2.5.2 Receiving a RopModifyRules ROP Request	Revised the product behavior note to state that Exchange 2013 does not enforce a limit on disabled rules.	Y	Product behavior note updated.
3.2.5.2 Receiving a RopModifyRules ROP Request	Added a product behavior note stating server behavior when changes push the rules beyond the limit.	Y	New product behavior note added.
3.2.5.2 Receiving a RopModifyRules ROP Request	Revised the description of the ecNotEnoughMemory error.	N	Content updated.

8 Index

T

V