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

Intellectual Property Rights Notice for Protocol Documentation

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

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

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

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

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

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

Preliminary

Copyright © 2008 Microsoft Corporation.

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

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

Preliminary

Copyright © 2008 Microsoft Corporation.

Table of Contents

Preliminary

Copyright © 2008 Microsoft Corporation.

Preliminary

Copyright © 2008 Microsoft Corporation.

1 Introduction

Rules are sets of conditions and associated actions that enable a user to automatically organize, categorize and act on messages as the messages are delivered to a folder.

This document specifies the E-mail Rules Protocol:

- The format in which a client can add, modify or delete rules on a folder
- The format in which a client can retrieve rules set on a folder
- Details that allow the server and the client to evaluate and execute rules

1.1 Glossary

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

address book

binary large object (BLOB)

Boolean

contents table

entry ID

FAI contents table

FAI message

folder

folder associated information (FAI)

folder ID (FID)

GUID

handle

little-endian

message

message ID (MID)

message object

named property

property

property ID

property tag

remote operation (ROP)

Preliminary

Copyright © 2008 Microsoft Corporation.

ROP request

ROP response

rule

store

special folder

table

Unicode

The following data types are defined in [MS-DTYP]:

BYTE

DWORD

WORD

ULONG

The following terms are specific to this document:

action: A discrete operation that is executed on an incoming **message** when all conditions in the same **rule** are TRUE. A **rule** contains one or more **actions**.

client-side rule: A **rule** that has at least one **action** that cannot be executed by the server and must be executed by the client.

condition: A logical expression comparing one or more properties in all incoming messages against a set of clauses. This logical expression can evaluate to TRUE or FALSE.

COUNT: A data type that is either a 2-byte **WORD** or a 4-byte **DWORD**, depending on the context where this data type is referenced: within a given buffer, **COUNT** is always 2 bytes or always 4 bytes, never a mix of the two.

Deferred Action Message (DAM): A hidden **message** indicating to the client it needs to execute one or more **rules** on another (user-visible) message in the **store**.

Deferred Error Message (DEM): A hidden **message** indicating to the client it needs to present the user with an error indicating a server-side **rule** failed to execute.

Deferred Action Folder (DAF): A **special folder** where the server places all **DAMs** and **DEMs** to be acted on by the client; this **folder** is not visible to the user.

extended rule: A **rule** that is added to, modified, and deleted from the server using a different mechanism than regular **rules** (**standard rules**), but is otherwise functionally identical to a **standard rule**. Content that applies only to extended **rules** is identified as such in the text of this document.

Preliminary

Copyright © 2008 Microsoft Corporation.

Out of Office rule: A **rule** that has the **ST_ONLY_WHEN_OOF** bit set in the PidTagRuleState **property**.

rule provider: A client application that created and maintains a specific **rule.** The application identifies itself using a unique, well-known string saved as a **property** on the **rule**.

Rule FAI Message: An **FAI message** stored in the **Inbox Special Folder** where the client can store extra rule-related information that is opaque to the server.

server-side rule: A **rule** for which all actions are executed by the server.

standard rule: A **rule** that is not an **extended rule**.

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

1.2 References

1.2.1 Normative References

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

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

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

[MS-OXCMSG] Microsoft Corporation, "Message and Attachment Object Protocol Specification", April 2008.

[MS-OXCPRPT] Microsoft Corporation, "Property and Stream Object Protocol Specification", April 2008.

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

[MS-OXCSTOR] Microsoft Corporation, "Store Object Protocol Specification", April 2008.

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

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

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

Preliminary

Copyright © 2008 Microsoft Corporation.

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

[MS-OXWOOF] Microsoft Corporation, "Out of Office (OOF) Web Service Protocol Specification", April 2008.

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

1.2.2 Informative References

None.

1.3 Protocol Overview (Synopsis)

The E-mail Rules Protocol specifies the client/server interaction that allows a messaging system to implement automatic message processing (message rules). This protocol documents a specific mechanism through which the server and the client can implement a flexible message processing system; since mail delivery is a complex operation, the server and the client may implement their own additional processing that is not covered by this protocol.

Rules are sets of conditions and associated actions 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 or private folders) <1>.

Rule 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 conditions in a rule are evaluated against the properties of the incoming message. If the conditions evaluate to TRUE, the rule actions are executed either by the server or by the client. If all actions in a rule can be executed by the server, the rule is said to be a **server-side rule**. If any action 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 moving messages to a personal store), the rule must 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 messaging client. Client-side rules do not execute until the mail 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 specified in [MS-OXOSFLD].

All (enabled) rules in a folder are evaluated in sequential order, one by one, until all rules in the rules table for the particular folder have been evaluated. If a particular rule's conditions are met, its associated set of actions is executed. If a rule is an "exit level" rule (according to a flag in the rule state property) and the rule condition is met, then the evaluation of subsequent

Preliminary

Copyright © 2008 Microsoft Corporation.

rules is cancelled. Otherwise, evaluation of the next rule continues even if a rule action moves the message, in which case the remaining rules continue to run against the moved message.

If the rule 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 (this is not common <1>), the server will evaluate the destination folder rules against the moved message after evaluating the remaining rules 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 the appropriate error when it processes the DEM.

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

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

The following sub-sections describe the main components covered in this protocol.

1.3.1 Creating, Modifying and Deleting Rules

Standard rules are created, modified and deleted using the ROPs specified in section 2.2.1 using the underlying [MS-OXCROPS] protocol.

Extended rules are created, modified, and deleted using an **FAI message** representation as specified in section 2.2.4 using the underlying [MS-OXCMSG] protocol.

1.3.2 Retrieving Rules from the Server

The messaging client can retrieve the standard rules in a folder in the form of a Table Object using the underlying ROP transport (see [MS-OXCROPS]) in the format specified in section 2.2.2.

Each row in the returned Table Object contains data representing one rule. The conditions, actions and other rule properties are returned as properties in the corresponding Table Row as specified in section 3.2.5.2.

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, as specified in section 2.2.4.1.

1.3.3 Executing Client-side Rules

When a rule cannot be executed entirely by the server, the client will need to complete the rule execution. The Rule Protocol specifies how this is achieved via Deferred Actions (section 3.1.4.1).

Preliminary

Copyright © 2008 Microsoft Corporation.

1.4 Relationship to Other Protocols

The [MS-OXORULE] protocol specification relies on an understanding of how to work with **folders**, **messages** and **tables** (for more detail see [MS-OXCMSG], [MS-OXCSTOR] and [MS-OXCTABL]). The specification also relies on understanding how ROPs defined in this protocol are transmitted to the server using the underlying transport (see [MS-OXCROPS]).

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

1.5 Prerequisites/Preconditions

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

1.6 Applicability Statement

The [MS-OXORULE] 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

None.

1.8 Vendor-Extensible Fields

A third party application can create its own set of rules by using its own custom string as the value of **PidTagRuleProvider** property (specified in section 2.2.1.3.2.5). There is no centralized authority that ensures uniqueness of Rule Provider strings across different client applications.

1.9 Standards Assignments

None.

Preliminary

Copyright © 2008 Microsoft Corporation.

2 Messages

2.1 Transport

The standard rules (sections 2.2.1, 2.2.2, and 2.2.3) are built using the ROP protocol specified in [MS-OXCROPS]. The extended rules portion of the protocol (section 2.2.4) is built using the messages protocol specified in [MS-OXCMSG].

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

2.2 Message Syntax

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

Before sending any of these requests to the server, the client MUST have successfully logged onto the server using **RopLogon**, and have a valid *LoginIndex* as specified in [MS-OXCROPS]. Also, the higher layers issuing the messages specified in this section MUST have opened **handles** to the **messaging objects** used as parameters in the ROP requests (as specified in each section).

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 Format

The messaging client sends the **RopModifyRules** request to create, modify or delete rules in a folder.

The syntax of the **RopModifyRules** request and response buffers are specified in the [MS-OXCROPS] protocol. This section specifies the syntax and semantics of various fields that are not fully specified in the Remote Operations (ROP) List and Encoding Protocol (see [MS-OXCROPS]).

2.2.1.1 Request Buffer

2.2.1.1.1 InputHandleIndex

The input handle for this operation is a folder object handle representing the folder for which rules are to be modified.

Preliminary

Copyright © 2008 Microsoft Corporation.

2.2.1.1.2 ModifyRulesFlag

This is a 8-bit field with a structure specified by the following table:

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 in this request are to replace existing rules in the folder; in this case, all subsequent RuleData structures MUST have ROW_ADD as the value of their RuleDataFlag field. If this bit is not set, the rules specified in this request represent changes (delete, modify, add) to the rules already existing in this folder.

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

2.2.1.1.3 RulesCount

This is a WORD field whose value MUST be the number of RuleData structures present in the ROP request.

2.2.1.1.4 RulesData Array

This is an array of **RuleData** structures that use the format specified in section 2.2.1.3

2.2.1.2 Response Buffer

2.2.1.2.1 InputHandleIndex

The input handle in the response buffer MUST be the same as the input handle in the request buffer for this operation.

2.2.1.2.2 ReturnValue

The ReturnValue is a 32-bit unsigned integer value that indicates the result of the operation. To indicate success, the server MUST return 0x00000000. For a list of common error return values, see [MS-OXCDATA].

2.2.1.3 RuleData Structure

The **RopModifyRules** request buffer MUST contain exactly ***RulesCount*** number of **RuleData** buffers, as specified in [MS-OXCROPS]. The following table specifies the format of the ***RuleData*** structures in the **RopModifyRules** request buffer.

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

Copyright © 2008 Microsoft Corporation.

Preliminary

RuleDataFlags	*PropertyValueCount*	*PropertyValue1* (variable)
PropertyValue1 (continued)		
…		
PropertyValueN (variable)		

RuleDataFlags (*1 byte*): MUST be set to one of the values specified in section 2.2.1.3.1.

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 exactly ***PropertyValueCount PropertyBuffer structures.

***PropertyValue1* (variable length structure): PropertyValue** structure containing one **property tag** and its associated value. The Property Tag used in this buffer MUST be among the ones specified in section 2.2.1.3.2. The format of the **PropertyValue** structure is specified in [MS-OXCDATA].

***PropertyValueN* (variable length structure):** Last of *PropertyValueCount* PropertyValue structures.

2.2.1.3.1 RuleDataFlags

The ***RuleDataFlags*** field in the **RuleData** structure MUST have one of the following values:

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

2.2.1.3.2 PropertyValue Structure

This section specifies the allowed set of property tags that MUST be used in the **PropertyValue** structure.

When deleting a rule, the only property the protocol client MUST pass is **PidTagRuleId** and SHOULD NOT pass in any other property. The protocol server MUST ignore properties other than **PidTagRuleId**. When adding a rule, the client MUST NOT pass in **PidTagRuleId**, it MUST pass in **PidTagRuleCondition, PidTagRuleActions** and

Copyright © 2008 Microsoft Corporation.

Preliminary

PidTagRuleProvider, and SHOULD <2> pass in the rest of the properties specified in this section. When modifying a rule, the client MUST pass in **PidTagRuleId** and SHOULD pass in the rest of the properties that need to be modified. The protocol server MUST ignore properties that are not specified in this section.

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

2.2.1.3.2.1 PidTagRuleId

An 64-bit unsigned integer value containing a unique identifier the messaging server generates for each rule when the rule is first created. The client MUST NOT specify this property when creating a new rule but MUST specify it when modifying or deleting a rule.

2.2.1.3.2.2 PidTagRuleSequence

A ULONG value used to determine the order in which rules are evaluated and executed. Rules are evaluated in sequence according to the increasing order of this value. The evaluation order for rules that have the same value in the **PidTagRuleSequence** property is undefined.

2.2.1.3.2.3 PidTagRuleState

A ULONG value interpreted as a bitmask combination of flags that specify the state of the rule. The value of **PidTagRuleState** is defined in the following table:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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, bit mask 0x00000001**):** The rule is enabled for execution. If this flag is not set, the server MUST skip this rule when evaluating rules.

ER (ST_ERROR, bit mask 0x00000002**):** The server has encountered an error processing the rule.

OF (ST_ONLY_WHEN_OOF, bit mask 0x00000004**):** The rule is executed only when the user sets the Out of Office state on the mailbox (see [MS-OXWOOF]. This flag MUST NOT be set in a Public Folder Rule. For details on this flag, see section 3.2

HI (ST_KEEP_OOF_HIST, bit mask 0x00000008**):** For details, see Out of Office Rule Processing in section 3.2. This flag MUST NOT be set in a Public Folder Rule.

EL (ST_EXIT_LEVEL, bit mask 0x00000010**):** Rule evaluation will terminate after executing this rule, except for evaluation of Out of Office rules.

Copyright © 2008 Microsoft Corporation.

Preliminary

SCL (ST_SKIP_IF_SCL_IS_SAFE, bit mask 0x00000020): Evaluation of this rule will be skipped if the delivered message's **PidTagContentFilterSCL** property has a value of 0xFFFFFFFF

PE (ST_RULE_PARSE_ERROR, bit mask 0x00000040): The server has encountered an error parsing the rule data provided by the client

X: Unused by this protocol. This bit MUST NOT be modified by the protocol client <3>

Note on the 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 condition evaluates to TRUE, and:

- the rule has the **ST_EXIT_LEVEL** flag set and does not have **ST_ONLY_WHEN_OOF** flag set, then the server MUST NOT evaluate subsequent rules that do not have **ST_ONLY_WHEN_OOF** flag set, and MUST evaluate subsequent rules that have **ST_ONLY_WHEN_OOF** flag set.

Or,

- the rule has both the **ST_EXIT_LEVEL** and **ST_ONLY_WHEN_OOF** flags set, then the server MUST NOT evaluate any subsequent rules.

2.2.1.3.2.4 PidTagRuleName

A string property that is the user-readable name of the rule.

2.2.1.3.2.5 PidTagRuleProvider

A string property identifying the client application that owns the rule. The client MUST specify this property when adding or modifying a rule.

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

A messaging client can define its own rule provider string, provided the value of the string is not the same as the rule provider string being used by another messaging client that could be setting rules on the same folder <4>.

2.2.1.3.2.6 PidTagRuleLevel

This property is not used; if setting this property, the client MUST pass in 0x00000000

2.2.1.3.2.7 PidTagRuleUserFlags

A LONG property set by the client for the exclusive use of the client. The server MUST preserve this value if it was set by the client, but MUST ignore it during rule evaluation and processing.

2.2.1.3.2.8 PidTagRuleProviderData

Copyright © 2008 Microsoft Corporation.

Preliminary

An opaque binary property that the client sets for the exclusive use of the client. The server MUST preserve this value if it was set by the client but MUST ignore its contents during rule evaluation and processing.

2.2.1.3.2.9 PidTagRuleCondition

The condition used when evaluating the rule. The condition is expressed as a Restriction (as specified in [MS-OXCDATA]) and the ***PropertyValue*** buffer contains the Restriction structure packaged as specified in [MS-OXCDATA] (using 2-byte sized COUNT values).

2.2.1.3.2.10 PidTagRuleActions

The set of actions associated with the rule. Its structure is specified in section 2.2.5, using a 2-byte sized COUNT value.

2.2.2 RopGetRulesTable Format

The syntax of the **RopGetRulesTable** request and response buffers are specified in the [MS-OXCROPS] protocol.

The **RopGetRulesTable** remote operation (ROP) creates a table object through which the client can access the standard rules in a folder using table operations as specified in the [MS-OXCTABL] protocol. The table returned by the server MUST contain all standard rules associated with a given folder. Each row in the table MUST represent one rule. The properties in each row are the properties set when the rule was created or modified.

This section specifies the syntax and semantics of various fields that are not fully specified in the [MS-OXCROPS] protocol.

2.2.2.1 Request Buffer

2.2.2.1.1 InputHandleIndex

The input handle for this operation is a folder object handle representing the folder for which rules are to be retrieved.

2.2.2.1.2 TableFlags

This is an 8-bit field as specified below.

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

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

Copyright © 2008 Microsoft Corporation.

Preliminary

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

2.2.2.2 Response Buffer

2.2.2.2.1 InputHandleIndex

The input handle in the response buffer MUST be the same as the input handle in the request buffer for this operation.

2.2.2.2.2 ReturnValue

The ReturnValue is a 32-bit unsigned integer value that indicates the result of the operation. To indicate success, the server MUST return 0x00000000. For a list of common error return values, see [MS-OXCDATA].

2.2.3 RopUpdateDeferredActionMessages Format

The format of the **RopUpdateDeferredActionMessages** request and response buffers are specified in the [MS-OXCROPS] protocol.

When the server generates DAMs for a message, the server MUST set the value of **PidTagHasDAMs** property on the message to TRUE.

During e-mail download, the **EntryId** identifying a message in the messaging system can change, for example when the client downloads the message into a store different than the server store. In such cases, as detailed in section 3.1.4.4, the messaging client MUST send the server a **RopUpdateDeferredActionMessages** request with the appropriate information on the message **EntryId** change as soon as the message's **EntryId** has been updated on the client.

This section specifies the syntax and semantics of various fields that are not fully specified in the [MS-OXCROPS] protocol.

2.2.3.1 Request Buffer

2.2.3.1.1 InputHandleIndex

The input handle for this operation is a logon object handle.

2.2.3.1.2 ServerEntryIdSize

WORD value representing the length, in bytes, of the ***ServerEntryId*** field

2.2.3.1.3 ServerEntryId

Byte array representing the EntryId of the message on the server for which the DAM has been generated. The length of this byte array is specified by the ***ServerEntryIdSize*** field.

2.2.3.1.4 ClientEntryIdSize

WORD value representing the length, in bytes, of the ***ClientEntryId*** field

Copyright © 2008 Microsoft Corporation.

Preliminary

2.2.3.1.5 ClientEntryId

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 Response Buffer

2.2.3.2.1 InputHandleIndex

The input handle in the response buffer MUST be the same as the input handle in the request buffer for this operation.

2.2.3.2.2 ReturnValue

The ReturnValue is a 32-bit unsigned integer value that indicates the result of the operation. To indicate success, the server MUST return 0x00000000. For a list of common error return values, see [MS-OXCDATA].

2.2.4 Extended Rules Message Syntax

The standard rules protocol has one major limitation as a consequence of using the ROP layer as the underlying transport – there is an inherent size limitation of 32kb per ROP package. To work around this limitation, extended rules were created. Extended rules are built using the [MS-OXCMSG] protocol. An extended rule is defined as an FAI message in a folder that MUST have the value of the **PidTagMessageClass** property set to "IPM.ExtendedRule.Message". This FAI message also has a set of rule-related properties set on it, as specified below. To create, modify or delete an Extended Rule, the application MUST create, modify or delete the underling FAI message.

Extended rules use a different set of properties than **RopModifyRules**. However, these properties map to properties for **RopModifyRules**; and except where noted, their formats are identical and the same syntactic restrictions and semantic meanings of values apply as the respective property defined in section 2.2.1.3.2.

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 MUST ignore any properties not explicitly listed here when evaluating extended rules.

2.2.4.1.1 PidTagRuleMsgName

This string property SHOULD be set on the FAI message. This property has the same semantics as **PidTagRuleName** defined in section 2.2.1.3.2.

2.2.4.1.2 PidTagMessageClass

Copyright © 2008 Microsoft Corporation.

Preliminary

This string property MUST be set on the FAI message, and MUST have a value of "IPM.ExtendedRule.Message".

2.2.4.1.3 PidTagRuleMsgSequence

This unsigned LONG property MUST be set on the FAI message. This property has the same semantics as **PidTagRuleSequence** defined in section 2.2.1.3.2.

2.2.4.1.4 PidTagRuleMsgState

This unsigned LONG property MUST be set on the FAI message. This property has the same semantics and flag meanings as **PidTagRuleState** defined in section 2.2.1.3.2.

2.2.4.1.5 PidTagRuleMsgUserFlags

This unsigned LONG property MAY be set on the FAI message. This property has the same semantics as **PidTagRuleUserFlags** defined in section 2.2.1.3.2.

2.2.4.1.6 PidTagRuleMsgLevel

This unsigned LONG property SHOULD be set on the FAI message. This property has the same semantics as **PidTagRuleLevel** defined in section 2.2.1.3.2.

2.2.4.1.7 PidTagRuleMsgProvider

This string property MUST be set on the FAI message. This property has the same semantics as **PidTagRuleProvider** defined in section 2.2.1.3.2.

2.2.4.1.8 PidTagRuleMsgProviderData

This binary property MAY be set on the FAI message. This property has the same syntax and semantics as **PidTagRuleProviderData** defined in section 2.2.1.3.2.

2.2.4.1.9 PidTagExtendedRuleMsgActions

This binary property MUST be set on the FAI message. This property serves the same purpose as **PidTagRuleActions**, 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 MUST be in Unicode format. The format of this property is defined in section 2.2.4.2.

2.2.4.1.10 PidTagExtendedRuleMsgCondition

This binary property MUST be set on the FAI message. This property serves the same purpose as **PidTagRuleCondition**, 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]), the client SHOULD keep the size of the PidTagExtendedRuleMsgCondition under the value specified by the **PidTagExtendedRuleSizeLimit** property; conversely, the

Copyright © 2008 Microsoft Corporation.

Preliminary

server SHOULD return an error if the client does attempt to set a binary property that is too large.

2.2.4.2 Extended Rule Actions Format

An extended rule's **PidTagExtendedRuleMsgActions** property contains additional information about the version of the rule and the named properties stored in the rule action, as well as information about the actions to be performed by this rule. The format of the binary property is specified in the following table:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NamedPropertyInformation (variable)																															
RuleVersion																															
RuleActionsBuffer (variable)																															

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

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

***RuleActionsBuffer* (variable length structure):** A structure containing the actions to be executed when this rule condition evaluates to true. The format of this structure is defined in section 2.2.5, using a 4-byte **COUNT** value (see Section 1.1 for a definition of **COUNT**).

2.2.4.3 Extended Rule Condition Format

Similar to extended rule actions, extended rule conditions contain information about any named properties contained inside of them. The format of the PidTagExtendedRuleMsgCondition binary property is specified in the following table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NamedPropertyInformation (variable)																															
RuleRestriction (variable)																															

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

Preliminary

Copyright © 2008 Microsoft Corporation.

***RuleRestriction* (variable length structure):** A structure containing the condition to be evaluated, represented as a **Restriction** structure. The format of this Restriction structure is defined in [MS-OXCDATA], using a 4-byte COUNT value.

2.2.4.4 Named Property Information Format

The named property information format provides context to any named property tags which are present in the structure it precedes. For every distinct (unique) named property used in the structure it precedes, the Named Property Information structure MUST contain one ***PropId*** – ***NamedProperty*** pair. Each ***PropId*** field is a **Property ID** that MUST have 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 in the following table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
NoOfNamedProps																*PropId-1*															
…																*PropId-N*															
NamedPropertiesSize																															
NamedProperty (variable)																															
...																															
NamedProperty-N (variable)																															

***NoOfNamedProps* (WORD):** 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.

PropId-1* (WORD):** the first ***PropId field

PropId-N* (WORD):** the last (NoOfNamedProps***) ***PropId*** field

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

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

NamedProperty-N* (variable length structure):** Specifies the last (NoOfNamedProps***) **PropertyName** structures.

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

Copyright © 2008 Microsoft Corporation.

Preliminary

2.2.5 Rule Action Format

The Rule Action data buffer MUST have one or more blocks of action block binary data to specify various actions of the rule, as specified in the following table.

The COUNT data type MUST be either a 2-byte WORD or a 4-byte DWORD, depending on the context where this section is referenced: within a given buffer, COUNT is always 2 bytes or always 4 bytes, never a mix of the two. Unless explicitly specified, any document referring to this section MUST assume a 2-byte size for the COUNT data type.

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-byte COUNT or 4-byte COUNT)																															
ActionBlock-1 (variable)																															
...																															
ActionBlock-N (variable)																															

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

***ActionBlock*-1 (variable length structure):** Specifies an action, see section 2.2.5.1.

ActionBlock*-N (variable length structure):** Specifies an action (the last of the ***NoOfActions ActionBlock fields).

Preliminary

Copyright © 2008 Microsoft Corporation.

2.2.5.1 Action Block Buffer Format

The format of action data block buffer is specified the following table:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ActionLength (2-byte COUNT or 4-byte COUNT)																															
ActionType																*ActionFlavor*															
ActionFlavor (continued)																*ActionFlags*															
ActionFlags (continued)																*ActionData* (variable)															

ActionLength* (COUNT):** MUST be the cumulative length (in BYTES) of the subsequent fields in this ***ActionBlock.

***ActionType* (BYTE):** Specifies the types of action (see table below)

ActionFlavor* (DWORD):** MUST be used in conjunction with specific ***ActionTypes that support it, and MUST be zero otherwise (see below)

***ActionFlags* (DWORD):** Client-defined flags.

ActionData* (variable length binary):** Specifies action 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:

Rule action types

Name	Value	Description
OP_MOVE	0x01	Moves the message to a folder. MUST NOT be used in a Public Folder Rule.
OP_COPY	0x02	Copies the message to a folder. MUST NOT be used in a Public Folder Rule.
OP_REPLY	0x03	Replies to the message.
OP_OOF_REPLY	0x04	Sends an Out of Office (OOF) reply to the message.

Preliminary

Copyright © 2008 Microsoft Corporation.

Name	Value	Description
OP_DEFER_ACTION	0x05	Used for actions that cannot be executed by the server (like playing a sound). MUST NOT be used in a Public Folder Rule.
OP_BOUNCE	0x06	Rejects the message back to the sender.
OP_FORWARD	0x07	Forwards the message to a recipient address.
OP_DELEGATE	0x08	Assigns the message to another recipient.
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 in the PidTagMessageFlags property on the message (see [MS-OXPROPS])

2.2.5.1.2 Action Flavors

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

If *ActionType* is **OP_FORWARD**, ***ActionFlavor*** MUST be a combination of the bitwise flags specified in the following table.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
X	X	X	X	X	AT	NC	PR	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

PR (bit mask 0x00000001): Preserves the sender information and indicates that the message was auto-forwarded. Can be combined with the NC ***ActionFlavor*** flag.

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

AT (bit mask 0x00000004): Makes the message an attachment to the forwarded 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.

Preliminary

Copyright © 2008 Microsoft Corporation.

If *ActionType* is **OP_REPLY** or **OP_OOF_REPLY**, ***ActionFlavor*** MUST have one of the values specified in the following table (a value of 0x00000000 indicates standard reply behavior):

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
X	X	X	X	X	X	ST	NS	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

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

ST: Server will use a fixed, server-defined text in the reply message and ignore the text in the reply template.

2.2.5.1.3 Action Data Buffer Format

The ***ActionData*** buffer is different for each ***ActionType*** 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 is used to move or copy an incoming message to a specified folder in the destination store. The following table specifies the ***ActionData*** buffer used in an action of type **OP_MOVE** or **OP_COPY**.

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
FolderInThisStore								*StoreEIDSize*																*StoreEID (variable)*							
StoreEID (continued)								*FolderEIDSize*																*FolderEID (variable)*							
FolderEID (continued)																															

FolderInThisStore* (BYTE):** MUST be either 0x01 if the folder whose Entry ID is ***FolderEID is in the protocol server store, or 0x00 if the folder is in a different store (for example, a local store the server cannot access).

***StoreEIDSize* (WORD):** The size of the *StoreEID* byte array

***StoreEID* (BYTE array):** The binary buffer specifies the destination store **EntryId**.

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

Copyright © 2008 Microsoft Corporation.

Preliminary

***FolderEID* (BYTE array):** The binary buffer specifies the destination folder's **EntryId**.

Furthermore, if the value of the ***FolderInThisStore*** field is 0x01, the ***FolderEID*** BYTE array MUST have the following structure:

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
Type								*FolderID*																							
...																															
...								*ID*																							
...																															
...								*Padding*																							
...																															

***Type* (BYTE):** MUST be set to 0x01

***FolderID* (8-byte unsigned integer):** MUST be set to the **FID** of the destination folder

***ID* (8-byte unsigned integer):** MUST be set to 0x0000000000000000

***Padding* (unsigned LONG):** MUST be set to 0x00000000

2.2.5.1.3.2 OP_REPLY and OP_OOF_REPLY Action Data Buffer Format

The following table specifies the Reply/OOF Reply *ActionData* buffer format.

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
ReplyTemplateFID																															
...																															
ReplyTemplateMID																															
...																															
ReplyTemplateGUID																															
...																															

Preliminary

Copyright © 2008 Microsoft Corporation.

…
…

***ReplyTemplateFID*:** The reply template 8-byte FID (see below)

***ReplyTemplateMID*:** The reply template 8-byte MID (see below)

***ReplyTemplateGUID*:** The reply template GUID (see below)

Before creating a rule that has an OP_REPLY or OP_OOF_REPLY ***ActionType***, the client MUST first create a Reply Template **FAI message** in the same folder as the rule. (For details on creating and manipulating **FAI messages**, see [MS-OXCMSG].)

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 to a string that has the prefix "IPM.Note.Rules.ReplyTemplate." (for OP_REPLY) or "IPM.Note.Rules.OOFTemplate." (for OP_OOF_REPLY)
3. Set the value of the **PidTagReplyTemplateId** property with a newly generated GUID
4. Set the value of **PidTagSubject** property, the text of the message and other message properties as desired.
5. Save the newly created message
6. Get the value of the **MID** and **FID** from the saved message

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

The ***ReplyTemplateGUID*** field in the reply ***ActionData*** buffer is the value of the GUID generated by the client at step 3 above, which is also stored on the reply template message as the value of the **PidTagReplyTemplateId** property. The ***ReplyTemplateGUID*** field MUST be unique in the folder - no two reply templates can share the same GUID.

2.2.5.1.3.3 OP_DEFER_ACTION Action Data Buffer Format

If one or more actions for a specific rule cannot be executed on the server, the rule MUST be a client-side rule, whose ***ActionType*** MUST be **OP_DEFER_ACTION**. Execution of the rule is postponed until the client is available.

The messaging client encodes the rule information as a client-dependent data structure designating the action 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 ***ActionLength*** value in the ***ActionBlock*** containing an **OP_DEFER_ACTION** *ActionType*.

Copyright © 2008 Microsoft Corporation.

Preliminary

If the action type is **OP_DEFER_ACTION**, the ***ActionData*** buffer is completely under the control of the messaging client that created the rule. This binary buffer MUST be treated as an opaque **binary large object (BLOB)** by the server. When a message that satisfies the rule condition is received, the server creates a DAM and places the entire content of the *ActionBlock* as a property on the DAM for the client to execute (see section 3.2.4.1.2).

2.2.5.1.3.4 OP_FORWARD and OP_DELEGATE Action Data Buffer Format

The following table specifies the ***ActionData*** buffer format that MUST be used with the OP_FORWARD and OP_DELEGATE action types.

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
RecipientCount (2-byte COUNT or 4-byte COUNT)																															
RecipientBlock-1 (variable)																															
…																															
RecipientBlock-N (variable)																															

***RecipientCount* (COUNT):** Specifies the number of recipient blocks. This number MUST be greater than zero.

***RecipientBlock*-1 (variable length binary):** Specifies recipient information (see table below)

RecipientBlock-N **(variable length binary):** Last of RecipientCount RecipientBlocks

The following table specifies the ***RecipientBlock*** data buffer.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved								*NoOfProperties* (2-byte COUNT or 4-byte COUNT)																								
NoOfProperties (continued)								*PropertyValue 1* (variable)																							
…																															
PropertyValue N (variable)																															

***Reserved* (BYTE):** MUST be set to 0x01.

Preliminary

Copyright © 2008 Microsoft Corporation.

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

***PropertyValue 1* (variable length structure):** Specifies the first **PropertyValue** structure.

PropertyValue N*(variable length structure):** Last of ***NoOfProperties **PropertyValue** structures

For details about parsing the PropertyValue structure, see [MS-OXCDATA]. The client MUST at a minimum specify values for the **PidTagDisplayName**, **PidTagEmailAddress**, and **PidTagRecipientType** properties in the forward/delegate ***ActionData*** buffer.

2.2.5.1.3.5 OP_BOUNCE Action Data Buffer Format

The following table shows the Bounce ***ActionData*** buffer format.

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
BounceCode																															

***BounceCode* (DWORD):** Specifies a bounce code as specified below.

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

Values	Description
0x0000000D	The message was refused because it was too large.
0x0000001F	The message was refused 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 **PropertyValue** structure, packaged as specified in [MS-OXCDATA].

2.2.5.1.3.7 OP_DELETE or OP_MARK_AS_READ Data Buffer Format

The incoming messages are deleted <5> or marked as read according to the *ActionType* itself. These actions have no ***ActionData*** buffer.

2.2.6 Deferred Action Message (DAM) Syntax

A Deferred Action Message MUST be created by the server to indicate to the client that it must further process a client-side rule action. This process is specified in section 3.2.4.1.2 of this document.

Copyright © 2008 Microsoft Corporation.

Preliminary

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

2.2.6.1 PidTagMessageClass

The value of this string property MUST be set to "IPC.Microsoft Exchange 4.0.Deferred Action"

2.2.6.2 PidTagDAMBackPatched

The value of this **Boolean** property 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.

2.2.6.3 PidTagDAMOriginalEntryId

The value of this binary property MUST be set to the EntryId of the delivered (target) message which the client has to process.

2.2.6.4 PidTagRuleProvider

The value of this string property MUST be set to the same value as the **PidTagRuleProvider** property on the rule(s) that have generated the DAM

2.2.6.5 PidTagRuleFolderEntryId

The value of this binary property MUST be set to the EntryId of the folder where the rule that triggered the generation of this DAM is stored

2.2.6.6 PidTagClientActions

The value of this binary property is a binary buffer specifying the actions the client must take on the message. The buffer MUST be packed according to the Rule Action Buffer format specified in section 2.2.5. The server MUST place in this property the relevant actions as they were set by the client when the rule was created or changed using **RopModifyRules**. Note that the server can combine actions from different rules into one DAM, in which case the rule actions will be concatenated in the DAM's **PidTagClientActions** using the proper action syntax specified in section 2.2.5.

2.2.6.7 PidTagRuleId

The value of this binary property is a binary buffer obtained by concatenating the **PidTagRuleId** values (8-bytes each) from all the rules that contributed actions in the **PidTagClientActions** property. The length of this binary property MUST be a multiple of 8 bytes.

Preliminary

Copyright © 2008 Microsoft Corporation.

2.2.7 Deferred Error Message (DEM) Syntax

A Deferred Error Message SHOULD be created by the server when an error is encountered while executing a rule. This process is specified in section 3.2.4.1.3 of this document.

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

2.2.7.1 PidTagMessageClass

The value of this string property MUST be set to "IPC.Microsoft Exchange 4.0.Deferred Error"

2.2.7.2 PidTagRuleError

The value of this unsigned LONG property MUST be set to one of the following values, indicating the cause of the error encountered during the execution of the rule:

0x00000001: generic error that doesn't fall into any of the other categories

0x00000002: error opening the rules folder

0x00000003: error delivering the message

0x00000004: error while parsing the rule format

0x00000005: error processing the rule

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 Deferred Action Message

0x00000009: error sending as another user

0x0000000A: error retrieving the reply template

0x0000000B: generic error while executing the rule on the server

0x0000000C: error processing rule due to mailbox quotas

0x0000000D: error processing the message due to the large number of recipients

0x0000000E: error copying or moving a message due to folder quotas

2.2.7.3 PidTagRuleActionType

The value of this unsigned LONG property MUST be set to the *ActionType* (see Rule Action Types in section 2.2.5.1.1) of the action in the rule that failed, or 0x00000000 if the failure is not specific to an action. Related property: **PidTagRuleActionNumber**.

Preliminary

Copyright © 2008 Microsoft Corporation.

2.2.7.4 PidTagRuleActionNumber

The value of this unsigned LONG property MUST be set to the index (0-based) of the action that failed, or 0x00000000 if the failure is not specific to an action; the *ActionType* of the action at this index MUST be the same value as the value of PidTagRuleActionType in this DEM. Related property: **PidTagRuleActionType**.

2.2.7.5 PidTagRuleProvider

The value of this string property MUST be set to the same value as the PidTagRuleProvider on the rule(s) that have caused the DEM to be generated

2.2.7.6 PidTagDAMOriginalEntryId

The value of this binary property MUST be set to the EntryId of the message that was being processed by the server when this error was encountered (i.e. the "delivered message")

2.2.7.7 PidTagRuleFolderEntryId

The value of this binary property MUST be set to the EntryId of the folder where the rule that triggered the generation of this DEM is stored.

2.2.7.8 PidTagRuleId

The value of this 8-byte unsigned integer property MUST be set to the same value as the value of the **PidTagRuleId** property on the rule that has generated this error.

Preliminary

Copyright © 2008 Microsoft Corporation.

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 sections describe the high-level objects used in this protocol:

3.1.1.1 Rules Table

When manipulating or using standard rules, the client may keep an in-memory table representation of the rules. The client ensures that its representation of the rules table and the server's representation match.

3.1.1.2 Deferred Actions Contents Table

The client maintains a contents table that describes the DAMs and DEMs contained in the Deferred Actions Folder (DAF). The client ensures that the rows in this table representing DAMs and DEMs are processed in a timely manner as specified in section 3.1.4.1.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Processing DAMs and DEMs

If the messaging client creates any rules, it SHOULD monitor the **Deferred Actions Folder (DAF)** for **DAMs** and **DEMs** the server may place in that folder and process the ones identified by the **PidTagRuleProvider** value the client supports. The server MUST place a message in the DAF either when it needs the client to perform an action as a result of a client-side rule (Deferred Action Message – DAM) or when it encounters a problem performing an action of a server-side rule (Deferred Error Messages – DEMs).

The DAF is a **Special Folder** that the server MUST create as specified in [MS-OXOSFLD].

Preliminary

Copyright © 2008 Microsoft Corporation.

After the client connects to the server, the higher layer SHOULD call the client implementing the [MS-OXORULE] protocol to inspect the contents of the DAF (as specified in [MS-OXCFOLD]). The higher layer in SHOULD also call the client when it detects a new DAM or DEM being placed in the DAF. The client MUST process DAMs and DEMs as specified in the subsections below.

3.1.4.1.1 Processing a DAM

When processing a DAM, the client MUST first determine if it has to process the DAM by inspecting the value of **PidTagRuleProvider** property on the DAM. If the value matches one of the Rule Provider strings the client supports, the client SHOULD process the DAM at its earliest convenience; otherwise the client MUST ignore the DAM.

In addition to **PidTagRuleProvider**, when processing a DAM the client can use any combination of the properties the server sets on the DAM as specified in section 3.2.1.3.1 in order to complete the execution of the rule. In particular, the client MUST use the value of **PidTagDAMOriginalEntryId** to identify the message it needs to take action on, and it SHOULD use the value of the **PidTagClientActions** property to identify what actions it needs to execute on the message.

After processing a DAM, the client MUST delete the DAM (for details about how to delete a message, see [MS-OXCFOLD]).

3.1.4.1.2 Processing a DEM

When processing a Deferred Error Message (DEM) the client MUST first determine if it has to process the DEM by inspecting the value of **PidTagRuleProvider** property 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 **PidTagRuleProvider**, when processing a DEM the client can use any combination of the properties the server sets on the DEM as specified in section 3.2.1.3.2. In particular, the client SHOULD use the value of **PidTagRuleError** to identify what error occurred, and it SHOULD use the values of **PidTagRuleFolderEntryId** and **PidTagRuleId** if it needs to get more information about the rule that failed and return that information to the higher levels.

As a result of processing the DEM, the higher levels SHOULD display an error to the user or take programmatic action as a result of a rule in error.

After processing a DEM, the messaging client MUST delete the DEM (for details about how to delete a message, see [MS-OXCFOLD]).

3.1.4.2 Retrieving Existing Rules

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

Copyright © 2008 Microsoft Corporation.

Preliminary

The table returned by **RopGetRulesTable** contains one rule per row. The columns available in this table are the properties specified in section 2.2.1.3.2, and their values are the same the client set previously using a **RopModifyRules** request. If the client did not set a value for that column property in the **RopModifyRules** request, the server MAY return an appropriate default value or it MAY return an error code in that column.

When a higher layer needs to inspect the extended rules or needs to display the extended rules to the user, the messaging client MUST retrieve the FAI contents table of the folder of interest and restrict the folder where PidTagMessageClass is equal to "IPM.ExtendedRules.Message". See [MS-OXCFOLD] and [MS-OXCTABL] for more details about retrieving an FAI contents table and restricting a table.

3.1.4.3 Adding, Modifying or Deleting Rules

When a higher layer needs to modify standard rules either or as a result of user interaction, the messaging client MUST do so using **RopModifyRules** as specified in section 2.2.1.

When the client adds or modifies rules, the client MUST ensure these operations do not affect existing rules that belong to different Rule Providers. The client also MUST ensure the uniqueness of the Rule Provider name it's using, i.e. the client MUST ensure that the value of the **PidTagRuleProvider** property it's using does not collide with well-known rule provider names used by other mail clients connecting to the same mailbox (for a list of well-known rule provider used by Outlook see section 2.2.1.3.2.5).

When adding a rule, the client MUST set values for **PidTagRuleProvider**, **PidTagRuleConditions** and **PidTagRuleActions** properties on each rule in the ROP request. The client MAY set values for **PidTagRuleUserFlags** and **PidTagRuleProviderData** as necessary for storing additional data. The client SHOULD send values for the other properties specified in section 2.2.1.3.2 in the ROP request as appropriate.

When modifying a rule, the client MUST send values for **PidTagRuleId** and SHOULD send values for properties that have changed, as specified in section 2.2.1.3.2.

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

The same considerations apply to equivalent (see section 2.2.4.1) properties present on an Extended Rule message. To add, modify, or delete an extended rule a messaging client would add, modify, or delete the FAI message representing that rule. The messaging client uses standard message operations to do so (refer to [MS-OXCMSG] for more details).

3.1.4.3.1 Public Folder Rules Considerations

The client MUST limit the conditions and actions that are available for Public Folders to server-side rules by only using rule actions that can be executed by the server.

3.1.4.3.2 Client-Specific Rule Metadata Storage

Preliminary

Copyright © 2008 Microsoft Corporation.

The client can choose to implement richer rules functionality than provided by the server (e.g. rules that are evaluated when sending a message). The client can also store additional rules metadata that is opaque to the server. If the client does have metadata associated with rules in the rules table, the client MUST store this metadata in a **Rules FAI Message** stored in the Inbox **Special Folder** (specified in [MS-OXOSFLD]).

The Rules FAI Message is an FAI message as specified in [MS-OXCMSG]. The client MUST create (or open, if already present) the Rules FAI Message in the Inbox **Special Folder**. This message MUST be identified by the values of its **PidTagSubject** and **PidTagMessageClass** 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 Rules FAI Message are completely up to the client application(s) and MUST be treated by the server as opaque [6].

3.1.4.4 Downloading a message to a different store

The messaging client can download or move a message from the server to a different store. As a result, the EntryId that uniquely identifies this message in the messaging system can change. Because the process of downloading or moving the message is independent of rules processing, the client MUST inform the server of this change in the value of the message EntryId.

If the client changes the EntryId of a message that has the **PidTagHasDAMs** property set to TRUE, the client MUST send a **RopUpdateDeferredActionMessages** to the server as specified in section 2.2.3, informing the server of the EntryId change.

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 and processed by the server. The client MUST process the ROP response associated with each message it sends.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

Preliminary

Copyright © 2008 Microsoft Corporation.

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.

3.2.1.1 Rules Table

The server MUST store all rules created or modified successfully by the client and return them to the client in the form of a rules table when requested. The server MUST also parse the rules set on each folder (according to the syntax specified in section 2.2.1) and evaluate and execute these rules when messages are delivered to that folder.

3.2.1.2 Out of Office State and Rule History List

While the mailbox is in an "Out of Office" state, for any rule that has the **ST_KEEP_OOF_HIST** flag set in **PidTagRuleState**, the server MUST keep a History List of all recipients for which the rule has fired once and MUST NOT execute the action again until the History List has been cleared.

The server MUST clear the History Lists for all rules when the mailbox exits the "Out of Office" state (as specified by [MS-OXWOOF]).

3.2.1.3 The Deferred Actions Folder (DAF)

The messaging server MUST create a DAF **Special Folder** as specified in [MS-OXSFLD] protocol. The DAF SHOULD support notifications on its contents table object (as specified in [MS-OXCFOLD] and [MS-OXCTABL]).

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

3.2.2 Timers

None.

3.2.3 Initialization

Prior to any client connecting to a mailbox, the server MUST ensure that the Deferred Action Folder (DAF) has been created for that mailbox according to the specification in [MS-OXOSFLD].

Preliminary

Copyright © 2008 Microsoft Corporation.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Message Delivery to a Folder

When a message is delivered to a private mailbox folder (or is posted to a public folder), the messaging server MUST <7> evaluate the rules that apply to the folder where the message was delivered. Note that it is possible for a rule to move the message to a folder where a different set of rules exist, in which case the server MUST apply rules recursively on the incoming message. Subsequent rules in the original folder MUST continue to execute on the moved message.

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

The server MUST only evaluate rules that are enabled, i.e. rules that have the **ST_ENABLED** flag set in the **PidTagRuleState** property.

The server MUST evaluate rules that have the **ST_ONLY_WHEN_OOF** flag set in the **PidTagRuleState** property only when the mailbox is in an "Out of Office" state as specified in [MS-OXWOOF].

When executing a rule whose condition evaluated to TRUE, the server MUST either perform the actions specified in that rule (in the case of a server-side rule) or generate a DAM for the client to process as specified in section 3.2.4.1.2. Here is what the server MUST do when it executes each action type 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 (e.g. body text properties, recipients on the template) and from the original message (e.g. the sender of the message) to create a reply to the message. The server MUST NOT send a reply if the **PidTagAutoResponseSuppress** property on the message has the 0x00000020 bit set. The server SHOULD also avoid sending replies to automatically generated messages and SHOULD avoid generating endless auto-reply loops.
- **OP_OOF_REPLY:** this is similar to the OP_REPLY action. In addition, the server MUST set the value of the **PidTagMessageClass** string property on the reply message to "IPM.Note.Rules.OofTemplate.Microsoft". The server MUST NOT send a reply if the **PidTagAutoResponseSuppress** property on the message has the 0x00000010 bit set.

Preliminary

Copyright © 2008 Microsoft Corporation.

- **OP_DEFER_ACTION:** the server MUST generate a DAM as specified in section 3.2.4.1.2. The server MUST also set the **PidTagHasDAMs** property to TRUE on the message.
- **OP_FORWARD:** the server MUST forward the message to the recipients specified in the action buffer structure002E
- **OP_DELEGATE:** the server MUST re-send the message to the recipients 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**:
    - PidTagReceivedRepresentingEntryId MUST be set to the same value as the mailbox user's PidTagEntryId
    - PidTagReceivedRepresentingAddrType MUST be set to the same value as the mailbox user's PidTagAddrType
    - PidTagReceivedRepresentingEmailAddress MUST be set to the same value as the mailbox user's PidTagEmailAddress
    - PidTagReceivedRepresentingName MUST be set to the same value as the mailbox user's PidTagDisplayName
    - PidTagReceivedRepresentingSearchKey MUST be set to the same value as the mailbox user's PidTagSearchKey
    - PidTagDelegatedByRule Boolean property MUST be set to TRUE
- **OP_BOUNCE:** the server MUST send a reply message to the sender detailing why it couldn't be delivered to the user's mailbox; the message MUST not appear in the user's mailbox (i.e. the message MUST be deleted)
- **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 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 on the message

If the server fails to execute a rule action, the server MUST generate a DEM as specified in section 3.2.4.1.3.

3.2.4.1.1 Out of Office Rules Processing

The server MUST evaluate and execute Out of Office rules only when the mailbox is in an "Out of Office" state (as specified in [MS-OXWOOF]). All Out of Office rules MUST be specified with the **ST_ONLY_WHEN_OOF** flag in the **PidTagRuleState** property as specified in section 2.2.1.3.2.3.

Preliminary

Copyright © 2008 Microsoft Corporation.

If a rule has the **ST_KEEP_OOF_HIST** flag set in the **PidTagRuleState** property, the server MUST check if the sender of the delivered message appears in the History List of recipients for that rule. If it does, the server MUST NOT evaluate the rule. If it doesn't and the rule condition evaluates to TRUE, the server MUST add the sender to the History List of recipients for that rule in addition to executing the rule action.

3.2.4.1.2 Generating a Deferred Action Message (DAM)

A server MUST generate a DAM when a rule condition evaluates to TRUE but the server cannot perform the actions specified in the rule.

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 may pack information about more than one OP_DEFER_ACTION actions for any given message into one DAM. The server SHOULD do this when there are more than one OP_DEFER_ACTION actions that belong to the same Rule Provider. The server MUST generate separate DAMs for OP_DEFER_ACTION actions that belong to separate Rules Providers.

3.2.4.1.3 Handling Errors During Rule Processing (creating a DEM)

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

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

Once 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 of that rule to prevent creating multiple DEMs with the same error information.

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], the server MUST start processing rules marked with the **ST_ONLY_WHEN_OOF** flag in the **PidTagRuleState** property. The server MUST also keep a Rule History List for rules that have the **ST_KEEP_OOF_HIST** flag in the **PidTagRuleState** property (see section 3.2.1.2).

When the mailbox exits the "Out of Office" state as specified in [MS-OXWOOF], the server MUST stop processing rules marked with the **ST_ONLY_WHEN_OOF** flag in the

Copyright © 2008 Microsoft Corporation.

Preliminary

PidTagRuleState property. The server MUST also clear the Rule History List for all rules (see section 3.2.1.2).

3.2.4.3 Server-side Rules Change

The server can choose to implement a User Interface that allows the user to modify all or some rules. Since the functionality provided by the mail client can exceed the functionality provided by the server as explained in section 3.1.4.3.2, if the server modifies any rules as a result of user interaction, the server MUST also delete the client-specific **Rules FAI Message** defined in section 3.1.4.3.2. The server SHOULD warn the user making the change that doing so might lead to loss of specific rule functionality implemented by the client.

3.2.5 Message Processing Events and Sequencing Rules

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

3.2.5.1 Processing RopModifyRules

When receiving a **RopModifyRules** request, 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 ***ReturnValue*** 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 present in the request buffer, the server MUST return 0x00000000 as the ***ReturnValue*** in the response buffer. The server MUST assign a value for the **PidTagRuleId** property for each rule that has been added by the **RopModifyRules** request. The value of **PidTagRuleId** on each rule MUST be unique in that folder. The server SHOULD also start using the newly modified rules when processing messages delivered to that folder as soon as it successfully processes the **RopModifyRules** request.

The server can limit the number of rules it allows a client to save on a folder. If the client attempts to create more rules than allowed by the server, the server MUST return an appropriate error ***ReturnValue*** code in the return buffer (see [MS-OXCROPS]).

The server MUST also update the value of the Boolean **PidTagHasRules** property when rules change on a folder. The value of this property MUST be set to TRUE if any rules are set in that folder and to FALSE otherwise.

3.2.5.2 Processing RopGetRulesTable

When receiving a **RopGetRulesTable** request, 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

Copyright © 2008 Microsoft Corporation.

Preliminary

buffer, or if any data in the request buffer is incorrect, the server MUST return an error ***ReturnValue*** in the response buffer.

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

3.2.5.3 Processing RopUpdateDeferredActionMessages

When receiving a **RopUpdateDeferredActionMessages** request, the server MUST parse the request according to the syntax specified in section 2.2.3. 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 ***ReturnValue*** in the response buffer.

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

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

Preliminary

Copyright © 2008 Microsoft Corporation.

4 Protocol Examples

Starting with a "clean" folder (i.e. a folder with no rules) here is a sample sequence of ROP requests and responses 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 which gets transmitted over the wire. Also note that the data for a multi-byte field appear in little-endian format, with the bytes in the field presented from least significant to most significant. Generally speaking, these ROP requests are packed with other ROP requests, compressed and packed in one or more RPC calls according to the specification in [MS-OXCROPS]. These examples assume the client has already successfully logged onto the server and opened the folder they wish to modify the rules on. For more details, see [MS-OXCROPS].

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

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

The bold value at the far left is the offset of the following bytes into the buffer, expressed in hexadecimal notation. Following the offset is a series of up to 16 bytes, with each two character sequence describing the value of one byte in hexadecimal notation. Here, the underlines byte "4d" (01001101) is located 0x83 bytes (131 bytes) from the beginning of the buffer. The dash between eighth byte ("44") and ninth byte ("4c") bytes has no semantic value, and serves only to distinguish the eight 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 it's 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 specified in [MS-OXCDATA]). The data appears in the buffer in little-endian format.

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

Property Tag: 0x66760003 (PidTagRuleSequence)

Property Value: 10

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

4.1 Adding a New Rule

In this example, the client wishes to add a rule to move emails to a folder named "X" when the subject contains the phrase "Project X". The client sends a **RopModifyRules** request, using the buffer format specified in section 2.2.1.

Preliminary

Copyright © 2008 Microsoft Corporation.

4.1.1 Client Request Buffer

A complete ROP request in this example would look like:

```
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 six bytes refer to the *RopId*, *LogonIndex*, *HandleIndex*, *ModifyRulesFlags*, and *RulesCount* fields of the **RopModifyRules** format as specified in [MS-OXCROPS].

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

ROP id: 0x41 (**RopModifyRules**)

LogonIndex: 0x00

Copyright © 2008 Microsoft Corporation.

Preliminary

HandleIndex: 0x01

ModifyRulesFlags: 0x00

RulesCount: 0x0001

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

0006: 01 08 00

RuleModificationFlag: 0x01 (ROW_ADD)

RulePropertyCount: 0x0008

The first of eight *RulePropertyBuffer*s begin at byte **0x0009**. They are summarized below. For more details on PropertyValue structure format, see [MS-OXCDATA].

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

Property Tag: 0x6682001f (**PidTagRuleName**)

Property Value: Unicode string: "Project X"

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

Property Tag: 0x66760003 (PidTagRuleSequence)

Property Value: 0x0000000a

0029: 03 00 77 66 01 00 00-00

Property Tag: 0x66770003 (PidTagRuleState)

Property Value: 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)

PropertyValue: RES_CONTENT condition, *FuzzyLevel* of 0x00010001 (FL_SUBSTRING | FL_IGNORECASE), where *PropertyTag* 0x0037001f (**PidTagSubject**) contains "Project X". For more details, see section 2.2.2.

Preliminary

Copyright © 2008 Microsoft Corporation.

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

Property Tag: 0x668000fe (PidTagRuleActions)

Property Value: 0x0001 actions, 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

Property Tag: 0x6681001f (PidTagRuleProvider)

Property Value: Unicode string: “RuleOrganizer”

014e: 03 00 83 66 00 00 00 00

Property Tag: 0x66830003 (PidTagRuleLevel)

Property Value: 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

Copyright © 2008 Microsoft Corporation.

Preliminary

Property Tag: 0x66840102 (PidTagRuleProviderData)

Property Value: Binary BLOB, 0x0010 bytes long, set by the client.

4.1.2 Server Responds to Client Request

A complete ROP response in this example would look like:

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

ROPid: 0x41 (RopModifyRules)

HandleIndex: 0x01

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

4.2 Displaying Rules to the User

In this example, the client wishes to display a list of active rules on a folder to the user. The client sends a **RopGetRulesTable** request, using the buffer format specified in section 2.2.2. The client also sends **RopSetColumns** and **RopQueryRows** requests, using the buffer format specified in [MS-OXCROPS] and [MS-OXCTABL].

4.2.1 Client Request for a Rules Table

A complete ROP request to request a rules table would look like:

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

ROPid: 0x3f (RopGetRulesTable)

LogonIndex: 0x00

HandleIndex: 0x00

RulesTableHandleIndex: 0x01

TableFlags: 0x40 (specifying a Unicode table)

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

In this case the client is interested in formatting the table and reading its rows, so the client also sends a **RopSetColumns** request:

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

ROPid: 0x12 (RopSetColumns)

Copyright © 2008 Microsoft Corporation.

Preliminary

LogonIndex: 0x00

HandleIndex: 0x01

WantAsync: 0x00 (Wait)

PropertyTagCount: 3

PropertyTag1: 0x66740014 (PidTagRuleId)

PropertyTag2: 0x66840102 (PidTagRuleProviderData)

PropertyTag3: 0x6682001f (PidTagRuleName)

The client also sends a **ROPQueryRows** request, to gather rows from the table:

0000: 15 00 01 00 01 32 00

ROPid: 0x15 (RopQueryRows)

LogonIndex: 0

HandleIndex: 1

WantCurrentRow: False (Advance)

WantForwardRead: True (forward reading)

RowCount: 50

In this example, the handle array at the end of the RPC contains:

0000: 23 02 00 00 ff ff ff ff

Handle Index 0: 0x00000223

Handle Index 1: 0xffffffff

Note that Handle Index 0 is referenced only in the **RopGetRulesTable** request – it refers to a table handle previously returned by **RopOpenFolder** (the inbox, for example). Handle Index 1 is referenced by the **RopGetRulesTable** (as the new rules table index), RopSetColumns (as the referenced table) and **RopQueryRows** (as the referenced table) 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 requests (**RopGetRulesTable**, **RopSetColumns**, **RopQueryRows**) and the server responds with three ROP responses:

0000: 3f 01 00 00 00 00

ROPid: 0x3f (RopGetRulesTable)

HandleIndex: 1

Preliminary

Copyright © 2008 Microsoft Corporation.

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)

HandleIndex: 1

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

CompletionStatus: 0x00 (TBLSTAT_COMPLETE)

The response to **RopQueryRows** 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 nine bytes of a **RopQueryRows** response contain data about the response:

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

ROPid: 0x15 (RopQueryRows)

HandleIndex: 1

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

Bookmark: 0x02 (BOOKMARK_END)

RowCount: 1

This is followed by the RowPropertyArray beginning at byte **0x0009**, which in this example contains one row (indicated by *RowCount* above). It is not possible to interpret this response without the context of the earlier RopSetColumns request as 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

HasError: False

Preliminary

Copyright © 2008 Microsoft Corporation.

Property1: 0x56f83f0100000001

Property2: 0x10 byte binary array

Property3: “Project X”

Property1, Property2, and Property3 correspond to PidTagRuleId, PidTagRuleProviderData, and PidTagRuleName as specified by the earlier **RopSetColumn** request. See [MS-OXCROPS] and [MS-OXCTABL] for more details.

At the end of the three ROP responses, the handle table is:

```
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 MUST use this handle for any further requests relating to the rules table.

4.3 Deleting a Rule

In this example, the client wishes to delete the rule created in section 4.1 using **RopModifyRules**. The client sends a **RopModifyRules** request, using the buffer format specified in section 2.2.1

4.3.1 Client Request Buffer

A complete ROP request in this example would look like:

```
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 six bytes refer to the *RopId*, *LogonIndex*, *HandleIndex*, *ModifyRulesFlags*, and *RulesCount* fields of the **RopModifyRules** format as specified in section 2.2.1.

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

ROPid: 0x41 (RopModifyRules)

LogonIndex: 0x00

HandleIndex: 0x00

ModifyRulesFlags: 0x00

RulesCount: 0x0001

Preliminary

Copyright © 2008 Microsoft Corporation.

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

0006: 04 01 00

RuleModificationFlag: 0x04 (ROW_REMOVE)

RulePropertyCount: x0001

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

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

PropertyTag: 0x66740014 (PidTagRuleId)

PropertyValue: 0x56f83f0100000001

4.3.2 Server Responds to Client Request

A complete ROP response in this example would look like:

0000: 41 00 00 00 00 00

ROPid: 0x41 (RopModifyRules)

HandleIndex: 0x00

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

Preliminary

Copyright © 2008 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the [MS-OXORULE] protocol. General security considerations pertaining to the underlying RPC-based transport apply (see [MS-OXCROPS]).

5.2 Index of Security Parameters

None.

6 Appendix A: Microsoft Office and Microsoft Exchange Behavior

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

Microsoft Office 2003 with Service Pack 3 applied

Microsoft Exchange 2003 with Service Pack 2 applied

Microsoft Office 2007 with Service Pack 1 applied

Microsoft Exchange 2007 with Service Pack 1 applied

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

<1> Section 1.3: Outlook 2003 and 2007 is only adding, modifying and deleting rules on the folders noted below. Outlook ignores rules set on any other folder(s).

- The Inbox **Special Folder** (for details, see [MS-OXOSFLD])
- Any public folder (for details, see [MS-OXCSTOR]) where the user has appropriate permissions; extended rules are not set or evaluated on public folders

<2> Depending on the type of rule, the Outlook client sometimes doesn't set all the properties that are noted as 'SHOULD' in this protocol. It is good practice however to fill in as many properties as possible.

Copyright © 2008 Microsoft Corporation.

Preliminary

<3> Exchange 2007 server implementation uses bit flags 0x00000080 and 0x00000100 to store information about Out of Office functionality; these bit flags are ignored by Outlook and by Exchange 2003 server. Bit flag 0x00000080 is used to disable a specific Out of Office rule on the Exchange 2007 server. Bit flag 0x00000100 has the same semantics as bit flag ST_ONLY_WHEN_OOF on the Exchange 2007 server. The rest of the flags are reserved for future use.

<4> Section 2.2.1.3.2.5: Outlook and Exchange define the following well-known Rule Provider strings; other rules clients MUST choose different strings when implementing different Rule Providers:

"MSFT:TDX Rules" (used by Public Folder Rules)

"MSFT:TDX OOF Rules" (used by Out of Office Rules in the Inbox)

"RuleOrganizer" + user defined string (user-defined rules in the Inbox Special Folder)

"Schedule+ EMS Interface" (used to assist with delegates)

"JunkEmailRule" (rule created to help with Junk Email filtering)

"MSFT:MR" (rule assisting the "Moderator" role on a Public Folder)

"MSFT: Public.Folder.FormRestrictions" (used by Public Folder Rules)

<5> Exchange performs a "hard delete" as specified in [MS-OXCFOLD], but this is not required for the protocol.

<6> Section 3.1.4.3.2: The Outlook mail client is using a stream property on the **Rules Associated Message** to store additional rule data that is opaque to the server. The Property Tag for this property is 0x68020102 (**PidTagRWRulesStream**). Other mail clients can use other opaque properties on the **Rules Associated Message** for storing client-specific rules data.

<7> Section 3.2.4.1: 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 only process the first two extended rules it encounters per folder. These settings are configurable by the administrator.

Preliminary

Copyright © 2008 Microsoft Corporation.

7 Index

Preliminary

Copyright © 2008 Microsoft Corporation.