Email Rules Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.

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

- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.

- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.

- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

**Support.** For questions and support, please contact dochelp@microsoft.com.
## Revision Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/4/2008</td>
<td>0.1</td>
<td>New</td>
<td>Initial Availability.</td>
</tr>
<tr>
<td>4/25/2008</td>
<td>0.2</td>
<td>Minor</td>
<td>Revised and updated property names and other technical content.</td>
</tr>
<tr>
<td>6/27/2008</td>
<td>1.0</td>
<td>Major</td>
<td>Initial Release.</td>
</tr>
<tr>
<td>8/6/2008</td>
<td>1.01</td>
<td>Minor</td>
<td>Updated references to reflect date of initial release.</td>
</tr>
<tr>
<td>9/3/2008</td>
<td>1.02</td>
<td>Minor</td>
<td>Revised and edited technical content.</td>
</tr>
<tr>
<td>12/3/2008</td>
<td>1.03</td>
<td>Minor</td>
<td>Minor editorial fixes.</td>
</tr>
<tr>
<td>4/10/2009</td>
<td>2.0</td>
<td>Major</td>
<td>Updated technical content and applicable product releases.</td>
</tr>
<tr>
<td>7/15/2009</td>
<td>3.0</td>
<td>Major</td>
<td>Revised and edited for technical content.</td>
</tr>
<tr>
<td>11/4/2009</td>
<td>4.0.0</td>
<td>Major</td>
<td>Updated and revised the technical content.</td>
</tr>
<tr>
<td>2/10/2010</td>
<td>5.0.0</td>
<td>Major</td>
<td>Updated and revised the technical content.</td>
</tr>
<tr>
<td>5/5/2010</td>
<td>5.0.1</td>
<td>Editorial</td>
<td>Revised and edited the technical content.</td>
</tr>
<tr>
<td>8/4/2010</td>
<td>6.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>11/3/2010</td>
<td>6.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>3/18/2011</td>
<td>7.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>8/5/2011</td>
<td>7.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>10/7/2011</td>
<td>7.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>1/20/2012</td>
<td>8.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>4/27/2012</td>
<td>9.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>7/16/2012</td>
<td>9.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>10/8/2012</td>
<td>10.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>2/11/2013</td>
<td>11.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>7/26/2013</td>
<td>12.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>11/18/2013</td>
<td>12.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>2/10/2014</td>
<td>12.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>4/30/2014</td>
<td>12.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>7/31/2014</td>
<td>12.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>Date</td>
<td>Revision History</td>
<td>Revision Class</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>----------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10/30/2014</td>
<td>12.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>3/16/2015</td>
<td>13.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>5/26/2015</td>
<td>14.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>9/14/2015</td>
<td>15.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>6/13/2016</td>
<td>16.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>9/14/2016</td>
<td>17.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>10/17/2016</td>
<td>17.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>7/24/2018</td>
<td>18.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>10/1/2018</td>
<td>19.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>12/11/2018</td>
<td>19.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>4/22/2021</td>
<td>20.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>8/17/2021</td>
<td>21.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>4/16/2024</td>
<td>22.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
</tbody>
</table>
# Table of Contents

## 1 Introduction ................................................................. 7
  1.1 Glossary ........................................................................... 7
  1.2 References ....................................................................... 10
    1.2.1 Normative References ..................................................... 10
    1.2.2 Informative References .................................................. 10
  1.3 Overview .......................................................................... 10
    1.3.1 Creating, Modifying, and Deleting Rules ....................... 11
    1.3.2 Retrieving Rules from the Server ................................. 11
    1.3.3 Executing Client-Side Rules ........................................ 12
  1.4 Relationship to Other Protocols ........................................ 12
  1.5 Prerequisites/Preconditions ............................................... 12
  1.6 Applicability Statement .................................................... 12
  1.7 Versioning and Capability Negotiation .............................. 12
  1.8 Vendor-Extensible Fields .................................................. 12
  1.9 Standards Assignments ..................................................... 12

## 2 Messages ............................................................................. 13
  2.1 Transport .......................................................................... 13
  2.2 Message Syntax ............................................................... 13
    2.2.1 RopModifyRules ROP .................................................... 13
      2.2.1.1 RopModifyRules ROP Request Buffer ....................... 13
      2.2.1.2 RopModifyRules ROP Response Buffer .................... 14
      2.2.1.3 RuleData Structure ................................................ 14
        2.2.1.3.1 Properties of a Standard Rule ............................. 14
          2.2.1.3.1.1 PidTagRuleId Property ................................. 15
          2.2.1.3.1.2 PidTagRuleSequence Property ..................... 15
          2.2.1.3.1.3 PidTagRuleState Property ............................ 15
          2.2.1.3.1.4 PidTagRuleName Property ............................ 16
          2.2.1.3.1.5 PidTagRuleProvider Property ...................... 16
          2.2.1.3.1.6 PidTagRuleLevel Property ............................ 16
          2.2.1.3.1.7 PidTagRuleUserFlags Property ..................... 16
          2.2.1.3.1.8 PidTagRuleProviderData Property ................ 16
          2.2.1.3.1.9 PidTagRuleCondition Property .................... 16
          2.2.1.3.1.10 PidTagRuleActions Property ..................... 16
      2.2.2 RopGetRulesTable ROP ................................................ 17
        2.2.2.1 RopGetRulesTable ROP Request Buffer ................. 17
        2.2.2.2 RopGetRulesTable ROP Response Buffer ................ 17
      2.2.3 RopUpdateDeferredActionMessages ROP ..................... 17
        2.2.3.1 RopUpdateDeferredActionMessages ROP Request Buffer 18
        2.2.3.2 RopUpdateDeferredActionMessages ROP Response Buffer 18
      2.2.4 Extended Rules Message Syntax ................................ 18
        2.2.4.1 Properties of an Extended Rule ............................ 18
          2.2.4.1.1 PidTagRuleMessageName Property .................. 18
          2.2.4.1.2 PidTagRuleMessageClass Property .................. 19
          2.2.4.1.3 PidTagRuleMessageSequence Property ............. 19
          2.2.4.1.4 PidTagRuleMessageState Property ................... 19
          2.2.4.1.5 PidTagRuleMessageUserFlags Property ............ 19
          2.2.4.1.6 PidTagRuleMessageLevel Property ................... 19
          2.2.4.1.7 PidTagRuleMessageProvider Property .............. 19
          2.2.4.1.8 PidTagRuleMessageProviderData Property ......... 19
          2.2.4.1.9 PidTagExtendedRuleMessageActions Property .... 20
          2.2.4.1.10 PidTagExtendedRuleMessageCondition Property 20
        2.2.4.2 NamedPropertyInformation Structure ..................... 21
      2.2.5 RuleAction Structure ................................................ 22
        2.2.5.1 ActionBlock Structure ........................................ 22
Email Rules Protocol

3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

3.1.1.1 Per Deferred Actions Contents Table

3.1.2 Timers

3.1.3 Initialization

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Retrieving Existing Rules

3.1.4.2 Adding, Modifying, or Deleting Rules

3.1.4.2.1 Adding, Modifying or Deleting Standard Rules

3.1.4.2.2 Adding, Modifying or Deleting Extended Rules

3.1.4.2.3 Creating Rules for Public Folders

3.1.4.2.4 Creating Rich Client-Side Rules

3.1.4.2.5 Creating a Reply Template

3.1.4.3 Downloading a Message to a Different Store

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Processing DAMs and DEMs

3.1.5.1.1 Processing a DAM

3.1.5.1.2 Processing a DEM

3.1.6 Timer Events

3.1.7 Other Local Events

3.2 Server Details
3.2.1 Abstract Data Model ................................................................. 38
  3.2.1.1 Per Mailbox ........................................................................ 38
  3.2.1.2 Per Message ..................................................................... 38
  3.2.1.3 Per Rules Table ................................................................. 38
  3.2.1.4 Per Rule ........................................................................... 38
3.2.2 Timers .................................................................................. 39
3.2.3 Initialization ........................................................................... 39
3.2.4 Higher-Layer Triggered Events ............................................... 39
  3.2.4.1 Returning and Maintaining the Rules Table ....................... 39
  3.2.4.2 Entering and Exiting the Out of Office State ..................... 39
3.2.5 Message Processing Events and Sequencing Rules .................. 39
  3.2.5.1 Processing Incoming Messages to a Folder ......................... 39
    3.2.5.1.1 Interaction Between ST_ONLY_WHEN_OOF and ST_EXIT_LEVEL Flags 41
    3.2.5.1.2 Generating a DAM .................................................... 42
    3.2.5.1.3 Handling Errors During Rule Processing (Creating a DEM) 42
  3.2.5.2 Receiving a RopModifyRules ROP Request ......................... 42
  3.2.5.3 Receiving a RopGetRulesTable ROP Request ..................... 43
  3.2.5.4 Receiving a RopUpdateDeferredActionMessages ROP Request 43
3.2.6 Timer Events ....................................................................... 44
3.2.7 Other Local Events .............................................................. 44

4 Protocol Examples ...................................................................... 45
  4.1 Adding a New Rule .................................................................. 45
  4.1.1 Client Request Buffer ........................................................ 45
  4.1.2 Server Responds to Client Request ...................................... 48
  4.2 Displaying Rules to the User .................................................... 48
    4.2.1 Client Request for a Rules Table ....................................... 48
    4.2.2 Server Responds to Client Requests ................................. 50
  4.3 Deleting a Rule ...................................................................... 51
    4.3.1 Client Request Buffer ....................................................... 51
    4.3.2 Server Responds to Client Request .................................... 52

5 Security .................................................................................... 53
  5.1 Security Considerations for Implementers ................................. 53
  5.2 Index of Security Parameters .................................................. 53

6 Appendix A: Product Behavior .................................................... 54

7 Change Tracking ........................................................................ 57

8 Index ........................................................................................ 58
1 Introduction

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

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

- **action**: (1) The smallest unit of work in a workflow system. An action can contain one or more tasks that define work that actors need to do. Actions are deployed and registered in the workflow system to be activated by protocol client users.

  (2) A discrete operation that is executed on an incoming **Message object** when all **conditions** in the same **rule** are TRUE. A rule contains one or more actions.

- **address book**: A collection of Address Book objects, each of which are contained in any number of address lists.

- **binary large object (BLOB)**: A discrete packet of data that is stored in a database and is treated as a sequence of uninterpreted bytes.

- **client-side rule**: A **rule** that has at least one **action** that is executed by a client because it cannot be executed by a server.

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

- **contents table**: A **Table object** whose rows represent the **Message objects** that are contained in a **Folder object**.

- **Deferred Action Folder (DAF)**: A special folder where a server places all Deferred Action Messages and Deferred Error Messages to be acted on by a client. The Deferred Action Folder is not visible to a user.

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

- **delegate**: A user or resource that has permissions to act on behalf of another user or resource.

- **entry ID**: See **EntryID**.

- **EntryID**: A sequence of bytes that is used to identify and access an object.

- **extended rule**: A **rule** that is added to, modified, and deleted from a server by using a mechanism other than standard rules, but is otherwise functionally identical to a standard rule.

- **FAI contents table**: A table of **folder associated information (FAI)** Message objects that are stored in a Folder object.

- **flags**: A set of values used to configure or report options or settings.

- **folder associated information (FAI)**: A collection of **Message objects** that are stored in a Folder object and are typically hidden from view by email applications. An FAI Message object is
used to store a variety of settings and auxiliary data, including forms, views, calendar options, favorites, and category lists.

**Folder object**: A messaging construct that is typically used to organize data into a hierarchy of objects containing Message objects and **folder associated information (FAI)** Message objects.

**globally unique identifier (GUID)**: A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the GUID. See also universally unique identifier (UUID).

**handle**: Any token that can be used to identify and access an object such as a device, file, or a window.

**hard delete**: A process that removes an item permanently from the system. If an item is hard deleted, a server does not retain a back-up copy of the item and a client cannot access or restore the item. See also soft delete.

**Inbox folder**: A special folder that is the default location for Message objects received by a user or resource.

**little-endian**: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

**Logon object**: A Server object that provides access to a private mailbox or a public folder. A client obtains a Logon object by issuing a RopLogon remote operation (ROP) to a server.

**mailbox**: A message store that contains email, calendar items, and other Message objects for a single recipient.

**Message object**: A set of properties that represents an email message, appointment, contact, or other type of personal-information-management object. In addition to its own properties, a Message object contains recipient properties that represent the addressees to which it is addressed, and an attachments table that represents any files and other Message objects that are attached to it.

**message store**: A unit of containment for a single hierarchy of Folder objects, such as a mailbox or public folders.

**named property**: A property that is identified by both a GUID and either a string name or a 32-bit identifier.

**Out of Office (OOF)**: One of the possible values for the free/busy status on an appointment. It indicates that the user will not be in the office during the appointment.

**Out of Office rule**: A rule that is only evaluated when the mailbox is in an Out of Office state.

**property ID**: A 16-bit numeric identifier of a specific attribute. A property ID does not include any property type information.

**property tag**: A 32-bit value that contains a property type and a property ID. The low-order 16 bits represent the property type. The high-order 16 bits represent the property ID.

**public folder**: A Folder object that is stored in a location that is publicly available.

**recipient**: (1) An entity that can receive email messages.
(2) An entity that is in an address list, can receive email messages, and contains a set of attributes. Each attribute has a set of associated values.

**remote operation (ROP):** An operation that is invoked against a server. Each ROP represents an action, such as delete, send, or query. A ROP is contained in a ROP buffer for transmission over the wire.

**remote procedure call (RPC):** A communication protocol used primarily between client and server. The term has three definitions that are often used interchangeably: a runtime environment providing for communication facilities between computers (the RPC runtime); a set of request-and-response message exchanges between computers (the RPC exchange); and the single message from an RPC exchange (the RPC message). For more information, see [C706].

**restriction:** A filter used to map some domain into a subset of itself, by passing only those items from the domain that match the filter. Restrictions can be used to filter existing Table objects or to define new ones, such as search folder or rule criteria.

**ROP request:** See ROP request buffer.

**ROP request buffer:** A ROP buffer that a client sends to a server to be processed.

**ROP response:** See ROP response buffer.

**ROP response buffer:** A ROP buffer that a server sends to a client to be processed.

**rule:** (1) A condition or action, or a set of conditions or actions, that performs tasks automatically based on events and values.

(2) An item that defines a condition and an action. The condition is evaluated for each Message object as it is delivered, and the action is executed if the new Message object matches the condition.

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

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

**rules table:** A Table object whose rows represent the rules that are contained in a Folder object.

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

**Short Message Service (SMS):** A communications protocol that is designed for sending text messages between mobile phones.

**special folder:** One of a default set of Folder objects that can be used by an implementation to store and retrieve user data objects.

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

**table object:** A group of shapes that are arranged in rows and columns to form a table.

**Unicode:** A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The Unicode standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

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

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

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

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

[MS-OXCMAIL] Microsoft Corporation, "RFC 2822 and MIME to Email Object Conversion Algorithm".


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

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


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

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


[MS-OXOMSG] Microsoft Corporation, "Email Object Protocol".

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


1.2.2 Informative References

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


1.3 Overview

The Email Rules Protocol enables the client/server interaction that allows a messaging system to implement automatic message processing (message rules (2)). This protocol provides a specific mechanism through which the server and the client can implement a flexible message processing system. Mail delivery is a complex operation that allows the server and the client to implement their own additional processing that is not covered by this protocol.
Rules (2) are sets of **conditions** and associated **actions (2)** that enable a user to automatically organize, categorize, and act on messages as the messages are delivered to a folder. Rules can be set on any server folder (either public folders or private folders).

Rule (2) evaluation is triggered when e-mail messages are delivered in a user's **mailbox** or when messages are first saved to a public folder. The clauses in a condition in a rule (2) are evaluated against the properties of the incoming message. If the condition evaluates to "TRUE", the rule (2) actions (2) are executed either by the server or by the client. If all actions (2) in a rule (2) can be executed by the server, the rule (2) is said to be a **server-side rule**. If any action (2) cannot be executed by the server (for example, the server doesn't have access to user's personal **message store**; therefore, it has to defer to the client any action (2) moving messages to a personal message store), the rule (2) has to be executed by the client, and it is said to be a **client-side rule**.

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

All enabled rules (2) in a folder are evaluated in sequential order, one by one, until all rules (2) in the **rules table** for the particular folder have been evaluated. If the conditions of a particular rule (2) are met, its associated set of actions (2) is executed. If a rule (2) is an "exit level" rule (2) (according to a flag in the rule (2) state property) and the rule (2) condition is met, then the evaluation of subsequent rules (2) is canceled. Otherwise, evaluation of the next rule (2) continues even if a rule (2) action (2) moves the message, in which case the remaining rules (2) continue to run against the moved message.

If the rule (2) 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 (2) (this is not common), the server will evaluate the destination folder rules (2) against the moved message after evaluating the remaining rules (2) in the original folder. If the destination folder does not exist, the server will create a **Deferred Error Message (DEM)** in the DAF, and the client will display an error when it processes the DEM.

When a folder is deleted, all rules (2) set on that folder are also deleted.

This protocol enables two slightly different types of rules (2): **standard rules**, which are more commonly used, and **extended rules**, which provide greater storage capacity, but for performance reasons, the server can choose to limit their usage. The way the two types of rules (2) are created and modified differs, but they are processed identically by the server and by the client.

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

### 1.3.1 Creating, Modifying, and Deleting Rules

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

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

### 1.3.2 Retrieving Rules from the Server

The client can retrieve the **standard rules** in a folder in the form of a **Table object**, as described in [MS-OXCTABL], by using the underlying **remote operation (ROP)** transport, as described in [MS-OXCROPS], in the format specified in section 2.2.2.
Each row in the returned Table object contains data representing one rule. The conditions, actions and other rule properties are returned as properties in the corresponding table row as specified in section 3.2.5.3.

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

1.3.3 Executing Client-Side Rules

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

1.4 Relationship to Other Protocols

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

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

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

1.5 Prerequisites/Preconditions

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

1.6 Applicability Statement

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

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

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

1.9 Standards Assignments

None.
2 Messages

2.1 Transport

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

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

2.2 Message Syntax

Standard rules are the most common and typical way of specifying rules (2) for a folder. Sections 2.2.1, 2.2.2, and 2.2.3 specify the ROP request buffers and ROP response buffers specific to this protocol. The syntax of these requests and responses is documented in [MS-OXCROPS], as specified in each section below.

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

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

2.2.1 RopModifyRules ROP

The RopModifyRules ROP ([MS-OXCROPS] section 2.2.11.1) creates, modifies, or deletes rules (2) in a folder.

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

2.2.1.1 RopModifyRules ROP Request Buffer

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

InputHandleIndex (1 byte): The index to the input handle for this operation, which is a Folder object handle representing the folder for which rules (2) are to be modified.

ModifyRulesFlags (1 byte): A bitmask that specifies how the rules (2) included in this structure are created on the server. Its structure is as follows.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>R</td>
</tr>
</tbody>
</table>

R (Bitmask 0x01): If this bit is set, the rules (2) in this request are to replace the existing set of rules (2) in the folder; in this case, all subsequent RuleData structures, as specified in section 2.2.1.3, MUST have the ROW_ADD flag as the value of their RuleDataFlags field, as specified in section 2.2.1.3.1. If this bit is not set, the rules (2) specified in this request represent changes (delete, modify, and add) to the set of rules (2) already existing in this folder.

x: Unused. This bit MUST be set to zero (0) when sent.
RulesCount (2 bytes): An integer that specifies the number of RuleData structures present in the RulesData field.

RulesData (variable): An array of RuleData structures, each of which specifies details about a standard rule. The format of the RuleData structure is specified in section 2.2.1.3.

2.2.1.2 RopModifyRules ROP Response Buffer

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

InputHandleIndex (1 byte): The input handle in the response buffer MUST be the same as the index to the input handle in the request buffer for this operation.

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

2.2.1.3 RuleData Structure

The RuleData structure contains properties and flags that provide details about a standard rule. The format of the RuleData structure is as follows.

```
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| RuleDataFlags | PropertyValueCount | PropertyValueCount | PropertyValueCount |
```

RuleDataFlags (1 byte): A value that contains flags specifying whether the rule (2) is to be added, modified, or deleted. The valid values are specified in the following table.

<table>
<thead>
<tr>
<th>Flag name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW_ADD</td>
<td>0x01</td>
<td>Adds the data in the rule buffer to the rule set as a new rule (2).</td>
</tr>
<tr>
<td>ROW_MODIFY</td>
<td>0x02</td>
<td>Modifies the existing rule (2) identified by the value of the PidTagRuleId property (section 2.2.1.3.1).</td>
</tr>
<tr>
<td>ROW_REMOVE</td>
<td>0x04</td>
<td>Removes from the rule set the rule (2) that has the same value of the PidTagRuleId property.</td>
</tr>
</tbody>
</table>

PropertyValueCount (2 bytes): An integer that specifies the number of properties that are specified in the PropertyValue field. This field MUST be greater than zero.

PropertyValues (variable): An array of TaggedPropertyValue structures, as specified in [MS-OXCDATA] section 2.11.4, each of which contains one property of a standard rule. This field MUST contain only properties that are valid for a standard rule, as specified in section 2.2.1.3.1. The number of TaggedPropertyValue structures contained in this field is specified in the PropertyValueCount field.

2.2.1.3.1 Properties of a Standard Rule

The properties for a standard rule are specified in sections 2.2.1.3.1 through 2.2.1.3.10. These properties are set by using the RopModifyRules ROP (section 2.2.1). The RuleData structure, which is specified in section 2.2.1.3, contains the property settings for each standard rule.
2.2.1.3.1.1 PidTagRuleId Property

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

The PidTagRuleId property ([MS-OXPROPS] section 2.950) specifies a unique identifier the messaging server generates for each rule (2) when the rule (2) is first created. The PidTagRuleId property MUST NOT be used when requesting that a new rule (2) be created but MUST be used when requesting that a rule (2) be modified or deleted.

2.2.1.3.1.2 PidTagRuleSequence Property

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

The PidTagRuleSequence property ([MS-OXPROPS] section 2.963) contains a value used to determine the order in which rules (2) are evaluated and executed. Rules (2) are evaluated in sequence according to the increasing order of this value. The evaluation order for rules (2) that have the same value in the PidTagRuleSequence property is undefined: the server can choose an arbitrary order for rules (2) with the same value, but that does not affect the sequence of other rules (2).

2.2.1.3.1.3 PidTagRuleState Property

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

The PidTagRuleState property ([MS-OXPROPS] section 2.964) contains a value interpreted as a Bitmask combination of flags that specify the state of the rule (2). The value of the PidTagRuleState property is defined as follows.

<table>
<thead>
<tr>
<th></th>
<th>ST_ENABLED</th>
<th>ST_ONLY_WHEN_OOF</th>
<th>ST_ERROR</th>
<th>ST_KEEP_OOF_HIST</th>
<th>ST_EXIT_LEVEL</th>
<th>ST_SKIP_IF_SCL_IS_SAFE</th>
<th>ST_RULE_PARSE_ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ER</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OF</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>HI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SCL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

EN (ST_ENABLED, Bitmask 0x00000001): The rule (2) is enabled for execution. If neither this flag nor the ST_ONLY_WHEN_OOF flag are set, the server skips this rule (2) when evaluating rules (2).

ER (ST_ERROR, Bitmask 0x00000002): The server has encountered any nonparsing error processing the rule (2). This flag is not to be set by the client and is to be ignored by the server if it is.

OF (ST_ONLY_WHEN_OOF, Bitmask 0x00000004): The rule (2) is executed only when a user sets the Out of Office (OOF) state on the mailbox, as specified in [MS-OXWOOF] section 2.2.5.2. This flag MUST NOT be set in a public folder rule (2). For details on this flag, see section 3.2.5.1.1.1.

HI (ST_KEEP_OOF_HIST, Bitmask 0x00000008): For details, see section 3.2.5.1.1. This flag MUST NOT be set in a public folder rule (2).

EL (ST_EXIT_LEVEL, Bitmask 0x00000010): Rule (2) evaluation will terminate after executing this rule (2), except for evaluation of Out of Office rules. For details, see section 3.2.5.1.1.1.

SCL (ST_SKIP_IF_SCL_IS_SAFE, Bitmask 0x00000020): Evaluation of this rule (2) MAY be skipped if the delivered message's PidTagContentFilterSpamConfidenceLevel property ([MS-OXPROPS] section 2.648) has a value of 0xFFFFFFFF.

PE (ST_RULE_PARSE_ERROR, Bitmask 0x00000040): The server has encountered rule (2) data from the client that is in an incorrect format, which caused an error parsing the rule (2) data. This flag is not to be set by the client and is to be ignored by the server if it is.
x: Unused by this protocol. Bit locations marked with x are to be set to 0, SHOULD NOT be modified by the client, and are ignored by the server.<1>

2.2.1.3.1.4 PidTagRuleName Property
Type: PtypString ([MS-OXCDATA] section 2.11.1)
The PidTagRuleName property ([MS-OXPROPS] section 2.960) specifies the name of the rule (2).

2.2.1.3.1.5 PidTagRuleProvider Property
Type: PtypString ([MS-OXCDATA] section 2.11.1)
The PidTagRuleProvider property ([MS-OXPROPS] section 2.961) identifies the client application that owns the rule (2). The client specifies this property when adding or modifying a rule (2).

Rules that are stored on folders are associated with the application that owns the rules (2) by using a rule provider string. Each client application is to only add, modify or delete rules (2) that it is responsible for.

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

2.2.1.3.1.6 PidTagRuleLevel Property
Type: PtypInteger32 ([MS-OXCDATA] section 2.11.1)
The PidTagRuleLevel property ([MS-OXPROPS] section 2.952) is not used; if a client requests that this property be set, the requested value MUST be 0x00000000.

2.2.1.3.1.7 PidTagRuleUserFlags Property
Type: PtypInteger32 ([MS-OXCDATA] section 2.11.1)
The PidTagRuleUserFlags property ([MS-OXPROPS] section 2.965) is an opaque property that the client sets for the exclusive use of the client. The server is to preserve this value if set by the client but ignores its contents during rule (2) evaluation and processing.

2.2.1.3.1.8 PidTagRuleProviderData Property
Type: PtypBinary ([MS-OXCDATA] section 2.11.1)
The PidTagRuleProviderData property ([MS-OXPROPS] section 2.962) is an opaque property that the client sets for the exclusive use of the client. The server is to preserve this value if set by the client but ignores its contents during rule (2) evaluation and processing.

2.2.1.3.1.9 PidTagRuleCondition Property
Type: PtypRestriction ([MS-OXCDATA] section 2.11.1)
The PidTagRuleCondition property ([MS-OXPROPS] section 2.947) sets the condition used when evaluating the rule (2). The condition is expressed as a restriction, as specified in [MS-OXCDATA] section 2.12.

2.2.1.3.1.10 PidTagRuleActions Property
Type: PtypRuleAction ([MS-OXCDATA] section 2.11.1)
The `PidTagRuleActions` property ([MS-OXPROPS] section 2.945) contains the set of actions (2) associated with the rule (2). Its structure is specified in section 2.2.5.

### 2.2.2 RopGetRulesTable ROP

The `RopGetRulesTable` ROP ([MS-OXCROPS] section 2.2.11.2) creates a Table object through which the client can access the standard rules in a folder using table operations as specified in [MS-OXCTABL]. The table returned by the server is required to contain all standard rules associated with a given folder. Each row in the table MUST represent one rule (2).

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

#### 2.2.2.1 RopGetRulesTable ROP Request Buffer

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

**InputHandleIndex (1 byte):** The index to the input handle for this operation, which is a Folder object handle representing the folder for which rules (2) are to be retrieved.

**TableFlags (1 byte):** The possible values for these bits are as follows.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>U</td>
</tr>
</tbody>
</table>

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

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

#### 2.2.2.2 RopGetRulesTable ROP Response Buffer

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

**OutputHandleIndex (1 byte):** The index to the output handle for this operation. MUST be set to the value of the `OutputHandleIndex` field specified in the request.

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

### 2.2.3 RopUpdateDeferredActionMessages ROP

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

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

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

**InputHandleIndex (1 byte):** The index to the input handle for this operation, which is a Logon object handle.

**ServerEntryIdSize (2 bytes):** An integer that specifies the length, in bytes, of the ServerEntryId field.

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

**ClientEntryIdSize (2 bytes):** An integer that specifies the length, in bytes, of the ClientEntryId field.

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

2.2.3.2 RopUpdateDeferredActionMessages ROP Response Buffer

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

**InputHandleIndex (1 byte):** The index to the input handle for this operation. This value MUST be the same as the index to the input handle in the request buffer for this operation.

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

2.2.4 Extended Rules Message Syntax

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

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

2.2.4.1 Properties of an Extended Rule

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

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

The **PidTagRuleMessageName** property ([MS-OXPROPS] section 2.954) SHOULD be set on the **FAI** message. This property has the same semantics as the **PidTagRuleName** property (section 2.2.1.3.1.4).

### 2.2.4.1.2 PidTagMessageClass Property

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

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

### 2.2.4.1.3 PidTagRuleMessageSequence Property

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

The **PidTagRuleMessageSequence** property ([MS-OXPROPS] section 2.957) MUST be set on the **FAI** message. This property has the same semantics as the **PidTagRuleSequence** property (section 2.2.1.3.1.2).

### 2.2.4.1.4 PidTagRuleMessageState Property

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

The **PidTagRuleMessageState** property ([MS-OXPROPS] section 2.958) MUST be set on the **FAI** message. This property has the same semantics and flag meanings as the **PidTagRuleState** property (section 2.2.1.3.1.3).

### 2.2.4.1.5 PidTagRuleMessageUserFlags Property

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

This **PidTagRuleMessageUserFlags** property ([MS-OXPROPS] section 2.959) MAY be set on the **FAI** message. This property has the same semantics as the **PidTagRuleUserFlags** property (section 2.2.1.3.1.7).

### 2.2.4.1.6 PidTagRuleMessageLevel Property

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

The **PidTagRuleMessageLevel** property ([MS-OXPROPS] section 2.953) SHOULD be set on the **FAI** message. This property has the same semantics as the **PidTagRuleLevel** property (section 2.2.1.3.1.6).

### 2.2.4.1.7 PidTagRuleMessageProvider Property

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

The **PidTagRuleMessageProvider** property ([MS-OXPROPS] section 2.955) MUST be set on the **FAI** message. This property has the same semantics as the **PidTagRuleProvider** property (section 2.2.1.3.1.5).

### 2.2.4.1.8 PidTagRuleMessageProviderData Property

Type: **PtypBinary** ([MS-OXCDATA] section 2.11.1)
The `PidTagRuleMessageProviderData` property ([MS-OXPROPS] section 2.956) MAY be set on the FAI message. This property has the same syntax and semantics as the `PidTagRuleProviderData` property (section 2.2.1.3.1.8).

### 2.2.4.1.9 PidTagExtendedRuleMessageActions Property

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

The `PidTagExtendedRuleMessageActions` property ([MS-OXPROPS] section 2.693) MUST be set on the FAI message. This property serves the same purpose for extended rules as the `PidTagRuleActions` property (section 2.2.1.3.1.10) serves for standard rules; however, it contains additional information about the version of the rule (2) and about the named properties used.

The format of the `PidTagExtendedRuleMessageActions` property is as follows.

```
 0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1
```

<table>
<thead>
<tr>
<th>NamedPropertyInformation (variable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>RuleVersion</td>
</tr>
<tr>
<td>RuleActionsBuffer (variable)</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

**NamedPropertyInformation (variable):** A structure that specifies information about named properties used in this action (2) as specified in section 2.2.4.2.

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

**RuleActionsBuffer (variable):** A `RuleAction` structure, as specified in section 2.2.5, containing the actions (2) to be executed when the condition for the rule (2) to which these actions (2) apply evaluates to "TRUE". All string values contained in any part of the `RuleAction` structure MUST be in Unicode format.

### 2.2.4.1.10 PidTagExtendedRuleMessageCondition Property

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

The `PidTagExtendedRuleMessageCondition` property ([MS-OXPROPS] section 2.694) MUST be set on the FAI message. This property serves the same purpose for extended rules as the `PidTagRuleCondition` property (section 2.2.1.3.1.9) serves for standard rules; 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. If the `PidTagExtendedRuleSizeLimit` property ([MS-OXCSTOR] section 2.2.2.1.1.1) is set on the Logon object, the client is required to keep the size of the `PidTagExtendedRuleMessageCondition` property under the value specified by the `PidTagExtendedRuleSizeLimit` property.

The format of the `PidTagExtendedRuleMessageCondition` property is as follows.
NamedPropertyInformation (variable): A structure that specifies information about named properties used in this condition, as specified in section 2.2.4.2.

RuleRestriction (variable): A structure containing the condition to be evaluated. The condition is expressed as a restriction, as specified in [MS-OXCDATA] section 2.12.

2.2.4.2 NamedPropertyInformation Structure

The NamedPropertyInformation structure provides context to any named properties that are present in the structure it precedes. For every distinct (unique) named property used in the structure it precedes, the NamedPropertyInformation structure contains one property ID – named property pair.

The format of the NamedPropertyInformation structure is as follows. Note that if there are no named properties to be listed, the NamedPropertyInformation structure reduces to a 2-byte value of 0x0000.

NamedPropertyInformation (variable)

...  

RuleRestriction (variable)

...  

NoOfNamedProps (2 bytes): An integer that specifies the number of named property mappings that are packed in this structure. If no named properties are used in the structure that follows the NamedPropertyInformation structure, the value of this field MUST be 0x0000 and no other fields are present.

PropIds (variable): An array of property IDs, each of which is a value of 0x8000 or greater and uniquely identifies the named property within an extended rule. There MUST be one property ID in this array for each(PropertyName) structure in the NamedProperties field.

NamedPropertiesSize (4 bytes): The total size, in bytes, of the following fields. Only present if NoOfNamedProps is greater than zero.
NamedProperties (variable): An array of PropertyName structures, each of which contains details about a named property. The format of the PropertyName structure is specified in [MS-OXCDATA] section 2.6.1. The PropertyName structures in this field MUST coincide with the property IDs in the PropIds field.

2.2.5 RuleAction Structure

The RuleAction structure MUST have one or more blocks of a binary data to specify various actions of the rule. The RuleAction structure has the following format for standard rules. The format for extended rules is the same except that the size of the NoOfActions field is 4 bytes instead of 2 bytes.

NoOfActions (2 bytes): Specifies the number of structures that are contained in the ActionBlocks field. This number MUST be greater than zero. For extended rules, the size of the NoOfActions field is 4 bytes instead of 2 bytes.

ActionBlocks (variable): An array of ActionBlock structures, each of which specifies an action of the rule, as specified in section 2.2.5.1.

2.2.5.1 ActionBlock Structure

The ActionBlock structure has the following format for standard rules. The format for extended rules is the same except that the size of the ActionLength field is 4 bytes instead of 2 bytes.

ActionLength (2 bytes): An integer that specifies the cumulative length, in bytes, of the subsequent fields in this ActionBlock structure. For extended rules, the size of the ActionLength field is 4 bytes instead of 2 bytes.

ActionType (1 byte): An integer that specifies the type of action. The valid actions are listed in the following table.

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

**ActionFlavor (4 bytes):** The flags that are associated with a particular type of action (2). The flags MUST be used in conjunction with the type of action (2) that supports them and MUST be zero otherwise. For more details see section 2.2.5.1.1.

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

**ActionData (variable):** An ActionData structure, as specified in section 2.2.5.1.2, that specifies data related to the particular action (2).

### 2.2.5.1.1 Action Flavors

The ActionFlavor field contains flags used in conjunction with the ActionType field and specifies additional information associated with the action (2) to be taken.

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

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

<table>
<thead>
<tr>
<th>ActionFlavor flag</th>
<th>Bitmask</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR</td>
<td>0x00000001</td>
</tr>
<tr>
<td>NC</td>
<td>0x00000002</td>
</tr>
<tr>
<td>AT</td>
<td>0x00000004</td>
</tr>
</tbody>
</table>

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

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

**AT (Bitmask 0x00000004):** Forwards the message as an attachment. This value MUST NOT be combined with other ActionFlavor flags.
TM (Bitmask 0x00000008): Indicates that the message SHOULD<5> be forwarded as a Short Message Service (SMS) text message. This value MUST NOT be combined with other ActionFlavor flags.

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

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

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| x | x | x | x | x | S | T | N | S | x | x | x | x | x | x | x | x | x | x | x | x |

NS (Bitmask 0x00000001): The server SHOULD NOT<6> send the message to the message sender (the reply template MUST contain recipients (2) in this case).

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

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

2.2.5.1.2 ActionData Structure

The ActionData structure is different for each type of action (2). The various ActionData structures are specified in section 2.2.5.1.2.1 through section 2.2.5.1.2.7. The appropriate structure MUST be used for the action (2) that is specified in the ActionType field of the ActionBlock structure.

2.2.5.1.2.1 OP_MOVE and OP_COPY ActionData Structure

A Move/Copy action (2) is used to move or copy an incoming message to a specified folder in the destination message store. The ActionData structure used in an action (2) of type "OP_MOVE" or "OP_COPY" is one of two formats, depending on whether the rule is a standard rule or an extended rule.

Buffer Format for Standard Rules

The OP_MOVE and OP_COPY ActionData structure MUST be in the following format for a standard rule. The destination folder for a Move/Copy action in a standard rule can be in the user's mailbox or a different mailbox.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| FolderInThisStore | StoreEIDSize | StoreEID (variable) |
| ... |
| FolderEIDSize | FolderEID (variable) |
| ... |

FolderInThisStore (1 byte): A Boolean value that indicates whether the folder is in the user's mailbox or a different mailbox. This field MUST be set to either a nonzero (TRUE) value, if the
destination folder is in the user's mailbox, or to zero (FALSE), if the destination folder is outside the user's mailbox (for example, a local message store that the server cannot access).

**StoreEIDSize (2 bytes):** An integer that specifies the size, in bytes, of the **StoreEID** field.

**StoreEID (variable):** A **Store Object EntryID** structure, as specified in [MS-OXCDATA] section 2.2.4.3, that identifies the message store. This field is relevant only if the **FolderInThisStore** field is set to 0x00.

If the **FolderInThisStore** field is set to 0x01, the following applies to the **StoreEID** field for both the client and the server:

- The **StoreEID** field can be set to any non-null value in request.
- The contents of the **StoreEID** field MUST be ignored when received.

**FolderEIDSize (2 bytes):** An integer that specifies the size, in bytes, of the **FolderEID** field.

**FolderEID (variable):** A structure that identifies the destination folder. If the value of the **FolderInThisStore** field is 0x01, this field contains a **ServerEid** structure, as specified in section 2.2.5.1.2.1.1. If the value of the **FolderInThisStore** field is 0x00, the contents of this field can be any value that the client finds useful for locating the destination folder when the client processes a **DAM** that it receives from the server.

**Buffer Format for Extended Rules**

The **OP_MOVE** and **OP_COPY** **ActionData** structure MUST be in the following format for an extended rule. The destination folder for a Move/Copy action in an extended rule MUST be in the user's mailbox.

| 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 |
| **StoreEIDSize**
| **StoreEID (variable)**
| ... 
| **FolderEIDSize**
| **FolderEID (variable)**
| ... 

**StoreEIDSize (4 bytes):** An integer that specifies the size, in bytes, of the **StoreEID** field.

**StoreEID (variable):** This field is not used and can be set to any non-null value by the client and the server. This field MUST be ignored when received by either the client or the server.

**FolderEIDSize (4 bytes):** An integer that specifies the size, in bytes, of the **FolderEID** field.

**FolderEID (variable):** A **Folder EntryID** structure, as specified in [MS-OXCDATA] section 2.2.4.1, that identifies the destination folder.

**2.2.5.1.2.1.1 ServerEid Structure**

The **ServerEid** structure contains details about the destination folder.
### Ours (1 byte)

The value 0x01 indicates that the remaining bytes conform to this structure; the value 0x00 indicates that the remaining bytes conform to a client-defined structure. This field MUST be set to 0x01.

### FolderId (8 bytes)

A Folder ID structure, as specified in [MS-OXCDATA] section 2.2.1.1, that identifies the destination folder.

### MessageId (8 bytes)

This field is not used and MUST be set to all zeros.

### Instance (4 bytes)

This field is not used and MUST be set to all zeros.

#### 2.2.5.1.2.2 OP_REPLY and OP_OOF_REPLY ActionData Structure

The ActionData structure used in an action (2) of type "OP_REPLY" or "OP_OOF_REPLY" is one of two formats, depending on whether the rule (2) is a standard rule or an extended rule. Prior to creating a rule (2) that has an action (2) of "OP_REPLY" or "OP_OOF_REPLY", the client is required to first create an FAI message to be used as the reply template. For details about creating a reply template, see section 3.1.4.2.5.

### Buffer Format for Standard Rules

The OP_REPLY and OP_OOF_REPLY ActionData structure MUST be in the following format for a standard rule.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| Ours | FolderId |
| ... |
| ... | MessageId |
| ... |
| ... | Instance |

<table>
<thead>
<tr>
<th>Ours</th>
<th>FolderId</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>MessageId</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Instance</td>
</tr>
</tbody>
</table>

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| ReplyTemplateFID |
| ... |
| ReplyTemplateMID |
| ... |
| ReplyTemplateGUID (16 bytes) |
| ... |
ReplyTemplateFID (8 bytes): A Folder ID structure, as specified in [MS-OXCDATA] section 2.2.1.1, that identifies the folder that contains the reply template.

ReplyTemplateMID (8 bytes): A Message ID structure, as specified in [MS-OXCDATA] section 2.2.1.2, that identifies the FAI message being used as the reply template.

ReplyTemplateGUID (16 bytes): A GUID that is generated by the client in the process of creating a reply template. The value of the ReplyTemplateGUID field is equal to the value of the PidTagReplyTemplateId property (section 2.2.9.2) that is set on the reply template.

Buffer Format for Extended Rules

The OP_REPLY and OP_OOF_REPLY ActionData structure MUST be in the following format for an extended rule.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| MessageEIDSize | 
| ReplyTemplateMessageEID (70 bytes) | ... |
| ... | 
| ... | 
| ReplyTemplateGUID (16 bytes) | ... |
| ... | 
| ... | 

MessageEIDSize (4 bytes): An integer that specifies the size, in bytes, of the ReplyTemplateMessageEID field.

ReplyTemplateMessageEID (70 bytes): A Message EntryID structure, as specified in [MS-OXCDATA] section 2.2.4.2, that contains the entry ID of the reply template.

ReplyTemplateGUID (16 bytes): A GUID that is generated by the client in the process of creating a reply template. The value of the ReplyTemplateGUID field is equal to the value of the PidTagReplyTemplateId property that is set on the reply template.

2.2.5.1.2.3 OP_DEFER_ACTION ActionData Structure

If one or more actions (2) for a specific rule (2) cannot be executed on the server, the rule (2) is required to be a client-side rule, with a value in the ActionType field of "OP_DEFER_ACTION". Execution of the rule (2) is postponed until the client is available.

The client encodes the rule (2) information as a client-dependent data structure designating the action (2) to be performed. The format is client-implementation-dependent and contains enough information to allow the client to perform the client-side operation when requested. The size of the buffer is
obtained by reading the value in the **ActionLength** field in the **ActionBlock** structure containing an **OP_DEFER_ACTION** in the **ActionType** field.

If the action (2) type is "OP_DEFER_ACTION", the **ActionData** structure is completely under the control of the client that created the rule (2). This structure MUST be treated as an opaque **BLOB** by the server. When a message that satisfies the rule (2) **condition** is received, the server creates a **DAM** and places the entire content of the **ActionBlocks** field of the **RuleAction** structure in the **PidTagClientActions** property (section 2.2.6.6) on the DAM as specified in sections 3.2.5.1.2, 2.2.6, and 2.2.6.6.

### 2.2.5.1.2.4 OP_FORWARD and OP_DELEGATE ActionData Structure

The **ActionData** structure that MUST be used with the "OP_FORWARD" and "OP_DELEGATE" action (2) types is formatted as follows.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| **RecipientCount** (2 bytes or 4 bytes) |  |
| **RecipientBlocks** (variable) |  |
| ... |  |

**RecipientCount (4 bytes)**: An integer that specifies the number of **RecipientBlockData** structures, as specified in section 2.2.5.1.2.4.1, contained in the **RecipientBlocks** field. This number MUST be greater than zero.

**RecipientBlocks (variable)**: An array of **RecipientBlockData** structures, each of which specifies information about one recipient (2).

### 2.2.5.1.2.4.1 RecipientBlockData Structure

The **RecipientBlockData** structure contains properties that specify information about a recipient (2). The client is required to, at a minimum, include the **PidTagDisplayName** ([MS-OXCOLD] section 2.2.2.2.2.5), **PidTagEmailAddress** ([MS-OXOABK] section 2.2.3.14), and **PidTagRecipientType** ([MS-OXOMSG] section 2.2.3.1) properties; some rules (2) MAY require more.

The **RecipientBlockData** structure has the following format.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
| **Reserved** | **NoOfProperties** |  |
| **...** | **PropertyValues (variable)** |  |
| **...** |  |  |

**Reserved (1 byte)**: This value is implementation-specific and not required for interoperability.<8>

**NoOfProperties (4 bytes)**: An integer that specifies the number of structures present in the **PropertyValues** field. This number MUST be greater than zero.
**PropertyValues (variable):** An array of TaggedPropertyValue structures, each of which contains a property that provides some information about the recipient (2). The format of the TaggedPropertyValue structure is specified in [MS-OXCDATA] section 2.11.4.

### 2.2.5.1.2.5 OP_BOUNCE ActionData Structure

The OP_BOUNCE ActionData structure is specified as follows.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3 | 0 | 1 |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | BounceCode |

**BounceCode (4 bytes):** An integer that specifies a bounce code.

The bounce code MUST be one of the following values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000000D</td>
<td>The message was rejected because it was too large.</td>
</tr>
<tr>
<td>0x0000001F</td>
<td>The message was rejected because it cannot be displayed to the user.</td>
</tr>
<tr>
<td>0x00000026</td>
<td>The message delivery was denied for other reasons.</td>
</tr>
</tbody>
</table>

### 2.2.5.1.2.6 OP_TAG ActionData Structure

An OP_TAG ActionData structure is a TaggedPropertyValue structure, packaged as specified in [MS-OXCDATA] section 2.11.4.

### 2.2.5.1.2.7 OP_DELETE or OP_MARK_AS_READ ActionData Structure

For the OP_DELETE or OP_MARK_AS_READ action types, the incoming messages are deleted or marked as read according to the ActionType itself. These actions (2) have no ActionData structure.

### 2.2.6 DAM Syntax

A DAM has to be created by the server to indicate to the client that it needs to further process a client-side rule action (2). This process is specified in section 3.2.5.1.2. **Extended rules** are not used in DAMs.

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

#### 2.2.6.1 PidTagMessageClass Property

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

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

#### 2.2.6.2 PidTagDamBackPatched Property

Type: PtypBoolean ([MS-OXCDATA] section 2.11.1)
The **PidTagDamBackPatched** property ([MS-OXPROPS] section 2.659) MUST be set to "FALSE" when the **DAM** is generated; it MUST be set to "TRUE" if the DAM was updated by the server as a result of a **RopUpdateDeferredActionMessages** request ([MS-OXCROPS] section 2.2.11.3).

### 2.2.6.3 PidTagDamOriginalEntryId Property

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

This **PidTagDamOriginalEntryId** property ([MS-OXPROPS] section 2.660) MUST be set to the **EntryID** of the delivered (target) message that the client has to process.

### 2.2.6.4 PidTagRuleProvider Property

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

The **PidTagRuleProvider** property ([MS-OXPROPS] section 2.961) MUST be set to the same value as the **PidTagRuleProvider** property on the rule or rules that have generated the **DAM**.

### 2.2.6.5 PidTagRuleFolderEntryId Property

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

The **PidTagRuleFolderEntryId** property ([MS-OXPROPS] section 2.949) MUST be set to the **EntryID** of the folder where the **rule** (2) that triggered the generation of this **DAM** is stored.

### 2.2.6.6 PidTagClientActions Property

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

The **PidTagClientActions** property ([MS-OXPROPS] section 2.634) is a binary buffer specifying the **actions** (2) the client is required to take on the message. The buffer MUST be packed according to the **RuleAction** structure specified in section 2.2.5.

The server is required to set values in this property according to the relevant actions (2) as they were set by the client when the **rule** (2) was created or changed by using the **RopModifyRules ROP** ([MS-OXCROPS] section 2.2.11.1). Note that the server can combine actions (2) from different rules (2) into one **DAM**, in which case the **RuleAction** structures will be concatenated in the **DAM**'s **PidTagClientActions** property by using the proper action (2) syntax specified in section 2.2.5.

### 2.2.6.7 PidTagRuleIds Property

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

The **PidTagRuleIds** property ([MS-OXPROPS] section 2.951) is a buffer that contains the **PidTagRuleId** (section 2.2.1.3.1.1) value (8 bytes) from the first **rule** (2) that contributed **actions** (2) in the **PidTagClientActions** property (section 2.2.6.6), and repeats that value once for each rule (2) that contributed actions (2). The length of this binary property MUST be a multiple of 8 bytes.

### 2.2.6.8 PidTagDeferredActionMessageOriginalEntryId Property

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

The **PidTagDeferredActionMessageOriginalEntryId** property ([MS-OXPROPS] section 2.662) contains the server **EntryID** for the **DAM** message on the server. This property is set by the server when the DAM is created.
2.2.7 DEM Syntax

A DEM SHOULD be created by the server when an error is encountered while executing a rule (2). This process is specified in section 3.2.5.1.3. Extended rules are not used in DEMs.

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

2.2.7.1 PidTagMessageClass Property

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

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

2.2.7.2 PidTagRuleError Property

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

The PidTagRuleError property ([MS-OXPROPS] section 2.948) MUST be set to one of the following values, indicating the cause of the error encountered during the execution of the rule (2).

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000001</td>
<td>Generic error that doesn't fall into any of the other categories.</td>
</tr>
<tr>
<td>0x00000002</td>
<td>Error opening the rules folder.</td>
</tr>
<tr>
<td>0x00000003</td>
<td>Error delivering the message.</td>
</tr>
<tr>
<td>0x00000004</td>
<td>Error while parsing the rule format.</td>
</tr>
<tr>
<td>0x00000005</td>
<td>Error processing the rule (2).</td>
</tr>
<tr>
<td>0x00000006</td>
<td>Error moving or copying the message to the destination folder.</td>
</tr>
<tr>
<td>0x00000007</td>
<td>Permission error moving or copying the message to the destination folder.</td>
</tr>
<tr>
<td>0x00000008</td>
<td>Error creating the DAM.</td>
</tr>
<tr>
<td>0x00000009</td>
<td>Error sending as another user.</td>
</tr>
<tr>
<td>0x0000000A</td>
<td>Error retrieving the reply template.</td>
</tr>
<tr>
<td>0x0000000B</td>
<td>Generic error while executing the rule (2) on the server.</td>
</tr>
<tr>
<td>0x0000000C</td>
<td>Error processing rule (2) due to mailbox quotas.</td>
</tr>
<tr>
<td>0x0000000D</td>
<td>Error processing the message due to the large number of recipients (2).</td>
</tr>
<tr>
<td>0x0000000E</td>
<td>Error copying or moving a message due to folder quotas.</td>
</tr>
</tbody>
</table>

2.2.7.3 PidTagRuleActionType Property

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

The PidTagRuleActionType property ([MS-OXPROPS] section 2.946) specifies the action (2) of the rule (2) that failed. This property MUST be set either to the value of the ActionType field, as
specified in section 2.2.5.1, or to 0x00000000 if the failure is not specific to an action (2). Related property: PidTagRuleActionNumber (section 2.2.7.4).

2.2.7.4 PidTagRuleActionNumber Property

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

The PidTagRuleActionNumber property ([MS-OXPROPS] section 2.944) MUST be set to the zero-based index of the action (2) that failed or set to 0x00000000 if the failure is not specific to an action (2). (For example, if specific to an action (2), a property value of 0x00000000 means that the first action (2) failed, 0x00000001 means that the second action (2) failed.) The ActionType field value of the action (2) at this index MUST be the same value as the value of the PidTagRuleActionType property (section 2.2.7.3) in this DEM.

2.2.7.5 PidTagRuleProvider Property

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

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

2.2.7.6 PidTagDamOriginalEntryId Property

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

The PidTagDamOriginalEntryId property (section 2.2.6.3) SHOULD<10> be set to the EntryID of the message that was being processed by the server when this error was encountered (that is, the "delivered message").

2.2.7.7 PidTagRuleFolderEntryId Property

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

The PidTagRuleFolderEntryId property (section 2.2.6.5) SHOULD<11> be set to the EntryID of the folder where the rule (2) that triggered the generation of this DEM is stored.

2.2.7.8 PidTagRuleId Property

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

The PidTagRuleId (section 2.2.1.3.1.1) property MUST be set to the same value as the value of the PidTagRuleId property on the rule (2) that has generated this error.

2.2.8 Rules-Related Folder Properties

2.2.8.1 PidTagHasRules Property

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

The PidTagHasRules property ([MS-OXPROPS] section 2.720) specifies whether rules (2) are set on a folder. This property SHOULD<12> be set to "TRUE" if any rules (2) are set on a folder and "FALSE" otherwise. If this property does not exist, it is treated as though its value is "FALSE".
2.2.9 Rules-Related Message Properties

2.2.9.1 PidTagHasDeferredActionMessages Property
Type: PtypBoolean ([MS-OXCDATA] section 2.11.1)
The PidTagHasDeferredActionMessages property ([MS-OXPROPS] section 2.718) specifies whether a message has at least one associated DAM. This property MUST be set to "TRUE" if it does and "FALSE" otherwise. If this property does not exist, it is treated as though its value is "FALSE".

2.2.9.2 PidTagReplyTemplateId Property
Type: PtypBinary ([MS-OXCDATA] section 2.11.1)
The PidTagReplyTemplateId property ([MS-OXPROPS] section 2.919) specifies the GUID for the reply template.

2.2.9.3 PidTagRwRulesStream Property
Type: PtypBinary ([MS-OXCDATA] section 2.11.1)
The PidTagRwRulesStream property ([MS-OXPROPS] section 2.966) contains client-specific data about the Rule FAI message. The value of this property is opaque to the server.
3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

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

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

Deferred Action Contents Table

3.1.1.1 Per Deferred Actions Contents Table

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

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Retrieving Existing Rules

When a higher layer needs to inspect the standard rules or needs to display these rules (2) to the user, the client MUST retrieve the rules (2) from the server using the RopGetRulesTable ROP request ([MS-OXCROPS] section 2.2.11.2) as specified in section 2.2.2. The higher level MUST use the returned table handle, as described in [MS-OXCTABL] section 1.5, to access rule (2) properties.

The table returned by the RopGetRulesTable ROP contains one rule (2) per row. The columns available in this table are the properties specified in section 2.2.1.3.1, and their values are the same as those the client set previously using a RopModifyRules ROP request ([MS-OXCROPS] section 2.2.11.1). If there isn't a value stored on the server for one of the rule (2) property columns, then when the client retrieves the rule (2) via a RopGetRulesTable ROP request, the server returns either a default value or an error for that column; which default values or errors are determined by the server implementation.

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

This section describes the process of adding, modifying or deleting rules (2).

3.1.4.2.1 Adding, Modifying or Deleting Standard Rules

When the client modifies standard rules as a result of user interaction, it MUST do so using a RopModifyRules ROP request ([MS-OXCROPS] section 2.2.11.1), as specified in section 2.2.1.<13>

When adding a standard rule, the client MUST NOT set a value for the PidTagRuleId property (section 2.2.7.8) and MUST set values for the PidTagRuleProvider (section 2.2.7.5), PidTagRuleCondition (section 2.2.1.3.1.9), and PidTagRuleActions (section 2.2.1.3.1.10) properties on each rule (2) in the ROP request buffer. The client MAY set values for the PidTagRuleUserFlags (section 2.2.1.3.1.7) and PidTagRuleProviderData (section 2.2.1.3.1.8) properties for storing additional data. The client SHOULD send values for the other properties specified in section 2.2.1.3.1 in the ROP request buffer.

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

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

3.1.4.2.2 Adding, Modifying or Deleting Extended Rules

To add, modify, or delete an extended rule, a client adds, modifies, or deletes the FAI message representing that rule (2) respectively. The client uses standard message operations, as specified in [MS-OXCMSG] section 3.1.4.

When adding an extended rule, the client MUST set values for the PidTagRuleMessageName (section 2.2.4.1.1), PidTagRuleMessageProvider (section 2.2.4.1.7), PidTagExtendedRuleMessageCondition, (section 2.2.4.1.10), and PidTagExtendedRuleMessageActions (section 2.2.4.1.9) properties for each rule (2) on the FAI message representing that rule (2). The client MAY set values for the PidTagRuleMessageUserFlags (section 2.2.4.1.5) and PidTagRuleMessageProviderData ([MS-OXPROPS] section 2.956) properties for storing additional data. The client SHOULD set values for the other properties on the FAI message, as specified in section 2.2.4.1.

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

When deleting an extended rule, the client MUST delete the FAI message representing that rule (2).

3.1.4.2.3 Creating Rules for Public Folders

When creating rules for public folders, the client MUST limit the conditions and actions (2) that are available for public folders to server-side rules by only using rule (2) actions (2) that can be executed by the server.

3.1.4.2.4 Creating Rich Client-Side Rules

To implement richer rules (2) functionality than provided by the server (for example, rules (2) that are evaluated when sending a message, the client can store additional rules (2) metadata that is opaque to the server. If the client does have metadata associated with rules (2) in the rules table, the client MUST store this metadata in a Rule FAI message stored in the Inbox folder. For more details about working with FAI messages, see [MS-OXCOLD] and [MS-OXCMSG].

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

Other properties on the Rule FAI message are up to the client application and MUST be treated by the server as opaque. The client uses the `PidTagRwRulesStream` property ([section 2.2.9.3](#)) on the Rule FAI message to store additional rule data. The client's use of other opaque properties on the Rule FAI message is determined by the implementer.

### 3.1.4.2.5 Creating a Reply Template

A reply template is a `FAI` message that is used when creating a reply message. Before creating a rule (2) that has an "OP_REPLY" or "OP_OOF_REPLY" value for the `ActionType` field, the client MUST first create a reply template in the folder for which the rule (2) is to be created.

The following steps specify how to create a reply template:

1. Create a new `FAI` message in the folder for which the rule (2) is to be created.
2. Set the value of the `PidTagMessageClass` property ([MS-OXCMMSG] section 2.2.1.3) to a string that has the prefix "IPM.Note.rules.ReplyTemplate." (for "OP_REPLY" values) or "IPM.Note.rules.OOFTemplate." (for "OP_OOF_REPLY" values).
3. Set the value of the `PidTagReplyTemplateId` property ([section 2.2.9.2](#)) with a newly generated `GUID`. This value MUST be unique in the folder—no two reply templates can share the same `GUID`.
4. Set the value of `PidTagSubject` property ([MS-OXCMMSG] section 2.2.1.46), the text of the message, and other message properties as desired.
5. Save the newly created message.
6. Get the value of the `Message ID` structure, as specified in [MS-OXCDATA] section 2.2.1.2, and `Folder ID` structure, as specified in [MS-OXCDATA] section 2.2.1.1, from the saved message.

The value of the `PidTagReplyTemplateId` property generated by the client at step 3 is the value used by the `ReplyTemplateGUID` field of the `OP_REPLY/OP_OOF_REPLY ActionData` structure specified in section 2.2.5.1.2.2.

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

### 3.1.4.3 Downloading a Message to a Different Store

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

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

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

3.1.5 Message Processing Events and Sequencing Rules

The messages specified in section 2.2 of this protocol are all sent by the client. The client processes the ROP response buffer associated with each message it sends as specified in section 2.2.1.2, section 2.2.2.2, and section 2.2.3.2. For more details on processing ROPs associated with rules (2), see [MS-OXCROPS] section 2.2.11.

3.1.5.1 Processing DAMs and DEMs

If the client creates any rules (2), the client SHOULD check the DAF for DAMs and DEMs placed in that folder and process the ones identified by the PidTagRuleProvider property value (section 2.2.1.3.1.5) the client supports. The DAF is a special folder that the server creates, as specified in section 3.2.1.3. The server places a message in the DAF either when it needs the client to perform an action (2) as a result of a client-side rule (DAM) or when it encounters a problem performing an action (2) of a server-side rule (DEM). When the server creates a DAM, it updates the PidTagDeferredActionMessageOriginalEntryId property (section 2.2.6.8), which is then used by the client in the ServerEntryId field of the RopUpdateDeferredActionMessages ROP request buffer (section 2.2.3).

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

3.1.5.1.1 Processing a DAM

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

In addition to the PidTagRuleProvider property, when processing a DAM, the client can use any combination of the properties the server sets on the DAM as specified in section 2.2.6 to execute the rule (2). In particular, the client MUST use the value of the PidTagDamOriginalEntryId property (section 2.2.6.3) to identify the message it needs to take action (2) on, and it SHOULD use the value of the PidTagClientActions property (section 2.2.6.6) to identify what actions (2) it needs to execute on the message.

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

3.1.5.1.2 Processing a DEM

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

In addition to the PidTagRuleProvider property, when processing a DEM, the client can use any combination of the properties the server sets on the DEM as specified in section 2.2.7 to execute the rule (2). In particular, the client SHOULD use the value of the PidTagRuleError property (section 2.2.7.2) to identify what error occurred, and it SHOULD use the values of the PidTagRuleFolderEntryId (section 2.2.7.7) and PidTagRuleId (section 2.2.7.8) properties if it needs to get more information from the rules table about the rule (2) that failed and return that information to the higher levels.
As a result of processing the DEM, the client SHOULD display an error to the user or take programmatic action (2) as a result of a rule (2) in error.

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

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

3.2.1 Abstract Data Model

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

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

Mailbox
Message
Rules Table
Rule

3.2.1.1 Per Mailbox

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

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

3.2.1.2 Per Message

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

3.2.1.3 Per Rules Table

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

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

3.2.1.4 Per Rule

A rule (4) is represented by the Rule ADM object type. The following ADM objects and states are maintained for each Rule ADM object type:
**Rule.Enabled**: True if this rule (2) is enabled and will execute when the conditions of the rule are met; otherwise, false.

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

**Rule.RuleData**: Specifies the conditions that trigger the rule (2).

### 3.2.2 Timers

None.

### 3.2.3 Initialization

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

### 3.2.4 Higher-Layer Triggered Events

#### 3.2.4.1 Returning and Maintaining the Rules Table

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

#### 3.2.4.2 Entering and Exiting the Out of Office State

When the mailbox enters the Out of Office state as specified in [MS-OXWOOF] section 2.2.4.2, the server MUST start processing rules (2) marked with the ST_ONLY_WHEN_OOF flag in the PidTagRuleState property (section 2.2.1.3.1.3). The server MUST also keep a list for rules (2) that have the ST_KEEP_OOF_HIST flag in the PidTagRuleState property specified in section 3.2.1.2.

When the mailbox exits the Out of Office state, the server MUST stop processing rules (2) marked with the ST_ONLY_WHEN_OOF flag in the PidTagRuleState property and clear the list for all rules (2). The semantics for processing rules (2) marked with the ST_ONLY_WHEN_OOF flag are specified in section 3.2.1.2.

### 3.2.5 Message Processing Events and Sequencing Rules

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

#### 3.2.5.1 Processing Incoming Messages to a Folder

When a message is either delivered to a private mailbox folder or posted to a public folder, the messaging server SHOULD evaluate the rules (2) that apply to the folder where the message was delivered. If a rule (2) moves the message to a folder where a different set of rules (2) exist, the server applies rules (2) recursively on the incoming message before executing any subsequent rules in the original folder.

A server can restrict the number of extended rules it executes on a folder. <14> The server can also restrict the maximum size, in bytes, that the user is allowed to accumulate for a single extended rule by setting the PidTagExtendedRuleSizeLimit property ([MS-OXCSTOR] section 2.2.2.1.1.1) on the...
Logon object. If the PidTagExtendedRuleSizeLimit property is set and the size of the PidTagExtendedRuleMessageCondition property (section 2.2.4.1.10) exceeds the value specified by the PidTagExtendedRuleSizeLimit property, the server MUST return an error.

For each message delivered to a folder, the server evaluates each rule (2) in that folder in increasing order of the value of the PidTagRuleSequence property (section 2.2.1.3.1.2) in each rule (2). If two or more rules (2) have the same value for the PidTagRuleSequence property, the order in which the server evaluates these rules (2) is not defined.

The server MUST only evaluate rules (2) that are enabled; that is, rules (2) that have the ST_ENABLED flag set in the PidTagRuleState property (section 2.2.1.3.1.3).

The server MUST evaluate rules (2) that have the ST_ONLY_WHEN_OOF flag set in the PidTagRuleState property only when the mailbox is in an OOF state as specified in [MS-OXWOOF] section 2.2.4.2.

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

- "OP_MOVE": The server MUST place a copy of the message in the folder specified in the action buffer structure and delete the original message; if multiple "OP_MOVE" operations apply to the same message, the server SHOULD create multiple copies of the message and then delete the original message.

- "OP_COPY": The server MUST place a copy of the message in the folder specified in the action buffer structure.

- "OP_REPLY": The server MUST use properties from the reply template (for example, body text properties, recipients (2) on the template) and from the original message (for example, the sender of the message) to create a reply to the message and then send the reply. The server MUST NOT send a reply if the PidTagAutoResponseSuppress property ([MS-OXOMSG] section 2.2.1.77) on the message has the 0x00000020 bit set. For more details on suppression of automatic replies, see [MS-OXOMAIL] section 2.2.3.2.14. The server MAY also avoid sending replies to automatically generated messages, which are identified by the PidTagAutoForwarded property ([MS-OXCMMSG] section 2.2.1.20), to avoid generating endless autoreply loops.

- "OP_OOF_REPLY": The server MUST behave as specified for the "OP_REPLY" action (2). In addition, the server SHOULD set the value of the PidTagMessageClass property ([MS-OXCMMSG] section 2.2.1.3) on the reply message to "IPM.Note.rules.OOFTemplate". This message class value is a prefix, and the client can append a client-specific value at the end; for example, a client can instead request that the server set the value of the PidTagMessageClass property in this circumstance to "IPM.Note.rules.OOFTemplate.Microsoft".<15> The server MUST NOT send a reply if the PidTagAutoResponseSuppress property on the message has the 0x00000010 bit set. For more information on suppression of automatic replies, see [MS-OXOMAIL] section 2.2.3.2.14.

- "OP_DEFER_ACTION": The server MUST generate a DAM as specified in section 3.2.5.1.2. The server MUST also set the PidTagHasDeferredActionMessages property (section 2.2.9.1) to "TRUE" on the message.

- "OP_FORWARD": The server MUST forward the message to the recipients (2) specified in the action buffer structure. The server SHOULD NOT<16> forward messages that were forwarded to the sender.

- "OP_DELEGATE": the server MUST resend the message to the recipients (2) specified in the action buffer structure. The server also MUST set the values of the following properties to match the current user's properties in the address book:
- The **PidTagReceivedRepresentingEntryId** property ([MS-OXOMSG] section 2.2.1.25) MUST be set to the same value as the mailbox user's **PidTagEntryId** property ([MS-OXOABK] section 2.2.3.3).

- The **PidTagReceivedRepresentingAddressType** property ([MS-OXOMSG] section 2.2.1.23) MUST be set to the same value as the mailbox user's **PidTagAddressType** property ([MS-OXOABK] section 2.2.3.13).

- The **PidTagReceivedRepresentingEmailAddress** property ([MS-OXOMSG] section 2.2.1.24) MUST be set to the same value as the mailbox user's **PidTagEmailAddress** property ([MS-OXOABK] section 2.2.3.14).

- The **PidTagReceivedRepresentingName** property ([MS-OXOMSG] section 2.2.1.26) MUST be set to the same value as the mailbox user's **PidTagDisplayName** property ([MS-OXCFOLD] section 2.2.2.2.2.5).

- The **PidTagReceivedRepresentingSearchKey** property ([MS-OXOMSG] section 2.2.1.27) MUST be set to the same value as the mailbox user's **PidTagSearchKey** property ([MS-OXCPRPT] section 2.2.1.9).

- The **PidTagDelegatedByRule** property ([MS-OXOMSG] section 2.2.1.84) MUST be set to "TRUE".

- "OP_BOUNCE": The server SHOULD send a reply message to the sender detailing why the sender's message couldn't be delivered to the user's mailbox; the original message SHOULD NOT appear in the user's mailbox.

- "OP_TAG": The server MUST set on the message the property specified in the action buffer structure.

- "OP_DELETE": The server MUST delete the message. The server MUST stop evaluating subsequent rules (2) on the message except for **Out of Office rules**.

- "OP_MARK_AS_READ": the server MUST set the **MSGFLAG_READ** flag (0x00000001) in the **PidTagMessageFlags** property ([MS-OXPROPS] section 2.792) on the message.

If the server fails to execute a rule (2) action (2), the server MUST generate a **DEM** as specified in section **3.2.5.1.3**.

The server MUST place all DAMs and DEMs that it creates as a result of running any rule (2) in any folder into the **DAF**.

### 3.2.5.1.1 Processing Out of Office Rules

The server evaluates and executes **Out of Office rules** only when the **mailbox** is in an **Out of Office** state, as specified in [MS-OXWOOF] section 2.2.4.2.

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

### 3.2.5.1.1.1 Interaction Between ST_ONLY_WHEN_OOF and ST_EXIT_LEVEL Flags

When the **Out of Office** state is set on the **mailbox**, as specified in [MS-OXWOOF], and a **rule (2)** condition evaluates to "TRUE", if the rule (2) has the **ST_EXIT_LEVEL flag** specified in section 2.2.1.3.1.3 set, then the server MUST NOT evaluate subsequent rules (2) that do not have the...
ST_ONLY_WHEN_OOF flag set. Subsequent rules (2) that have the ST_ONLY_WHEN_OOF flag set MUST be evaluated.

3.2.5.1.2 Generating a DAM

A server MUST generate a DAM when a rule (2) condition evaluates to "TRUE" but the server cannot perform the actions (2) specified in the rule (2). When the server generates DAMs for a message, the server MUST set the value of the PidTagHasDeferredActionMessages property (section 2.2.9.1) on the message to "TRUE".

The server MUST generate the DAM in the following manner:

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

The server can pack information about more than one "OP_DEFER_ACTION" actions (2), as specified in section 2.2.5.1.1, for any given message into one DAM. The server SHOULD do this when there are more than one "OP_DEFER_ACTION" actions (2) that belong to the same rule provider. The server MUST generate separate DAMs for "OP_DEFER_ACTION" actions (2) that belong to separate rule providers.

3.2.5.1.3 Handling Errors During Rule Processing (Creating a DEM)

A server SHOULD generate a DEM when it encounters an error processing a rule (2) on an incoming message. The server SHOULD also generate a DEM if it fails to create a DAM for a specific rule (2).

The server MUST generate the DEM in the following manner:

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

The first time the server finds a server-side rule to be in error and has generated a DEM for it, the server MUST set the ST_ERROR flag in the PidTagRuleState property (section 2.2.1.3.1.3) of that rule (2). Examination of the ST_ERROR flag on subsequent operations is used to prevent creating multiple DEMs with the same error information.

3.2.5.2 Receiving a RopModifyRules ROP Request

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

If the server successfully parses the data in the request buffer and is able to process all requests for adding, modifying, and deleting rules (2) present in the request buffer, the server MUST return 0x00000000 as the value of the ReturnValue field in the response buffer. The server MUST assign a value for the PidTagRuleId property (section 2.2.7.8) for each rule (2) that has been added by the RopModifyRules ROP request. The value of the PidTagRuleId property on each rule (2) MUST be unique in that folder.

The server can limit the rules (2) it allows on a folder to a certain number of rules (2) or to a total aggregate size of rules. If a RopModifyRules request causes the rules (2) to exceed the limit, the server MUST return the ecNotEnoughMemory (0x8007000E) error in the ReturnValue field of the
RopModifyRules response. Regardless of the limit, the server SHOULD save all changes specified by the RopModifyRules request.

The server MUST update the value of the PidTagHasRules property (section 2.2.8.1) when rules (2) change on a folder. The value of this property MUST be set to "TRUE" if any rules (2) are set in that folder and to "FALSE" otherwise. The server SHOULD start using the newly modified rules (2) when processing messages delivered to that folder as soon as it successfully processes the RopModifyRules ROP request. Any rules that exceed the limit are disabled during message delivery to the folder.

The following error codes can be returned by this ROP.

<table>
<thead>
<tr>
<th>Error code name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecInvalidParam</td>
<td>0x80070057</td>
<td>One or more of the x bits in the ModifyRulesFlag field of the ROP request is not set to 0.</td>
</tr>
<tr>
<td>ecNotEnoughMemory</td>
<td>0x8007000E</td>
<td>The number of rules (2) has been exceeded or the aggregate size of rules (2) has been exceeded.</td>
</tr>
</tbody>
</table>

### 3.2.5.3 Receiving a RopGetRulesTable ROP Request

When receiving a RopGetRulesTable ROP request ([MS-OXCROPS] section 2.2.11.2), the server MUST parse the request according to the syntax specified in section 2.2.2. If the server encounters an error parsing the request buffer, or if any data in the request buffer is incorrect, the server MUST return an error in the ReturnValue field of the ROP response buffer. A list of common error return values are described in [MS-OXCDATA] section 2.4.

If the server successfully parses the data in the ROP request buffer, it MUST return 0x00000000 as the value of the ReturnValue field in the response buffer and MUST return a valid table handle through which the client can access the folder rules (2) using table specific ROPs defined in [MS-OXCTABL].

The following error code can be returned by this ROP.

<table>
<thead>
<tr>
<th>Error code name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecNotSupported</td>
<td>0x80040102</td>
<td>One or more of the x bits on the TableFlags field is set to a nonzero value.</td>
</tr>
</tbody>
</table>

### 3.2.5.4 Receiving a RopUpdateDeferredActionMessages ROP Request

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

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

### 3.2.6 Timer Events

None.

### 3.2.7 Other Local Events

None.
4 Protocol Examples

Starting with a "clean" folder (that is, a folder with no rules (2)), here is a sample sequence of ROP request buffers and ROP response buffers that a client and a server might exchange. Note that the examples listed here only show the relevant portions of the specified ROPs; this is not the final byte sequence that gets transmitted over the wire. Also note that the data for a multibyte field appear in little-endian format, with the bytes in the field presented from least significant to most significant. Generally speaking, these ROP request buffers are packed with other ROP request buffers, compressed and packed in one or more remote procedure calls (RPCs) as described in [MS-OXCROPS]. These examples assume the client has already successfully logged on to the server and opened the folder on which it will modify the rules (2).

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

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

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

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

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

0021: 03 00 76 66 0a 00 00 00

PropertyTag: 0x66760003 (PidTagRuleSequence (section 2.2.1.3.1.2))

PropertyValue: 10

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

4.1 Adding a New Rule

In this example, a user wants to add a rule (2) to move e-mail messages to a folder named "X" when the subject contains the phrase "Project X". The client sends a RopModifyRules ROP request ([MS-OXCROPS] section 2.2.11.1), in the buffer format specified in section 2.2.1.

4.1.1 Client Request Buffer

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

0000: 41 00 01 00 01 00 01 08~00 1f 00 82 66 50 00 72
0010: 00 6f 00 6a 00 65 00 63 00 20 00 58 00 00

[MS-OXORULE] - v20240416
Email Rules Protocol
Copyright © 2024 Microsoft Corporation
Release: April 16, 2024
The first 6 bytes refer to the **RopId**, **LogonId**, **InputHandleIndex**, **ModifyRulesFlags**, and **RulesCount** fields of the **RopModifyRules** format as described in [MS-OXCROPS] section 2.2.11.1.

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

**RopId**: 0x41 (**RopModifyRules**)

**LogonId**: 0x00

**InputHandleIndex**: 0x01

**ModifyRulesFlags**: 0x00

**RulesCount**: 0x0001

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

```
0006: 01 08 00
```

**RuleDataFlags**: 0x01 (**ROW_ADD**)

**PropertyValueCount**: 0x0008

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

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

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

**PropertyValue**: **Unicode** string: "Project X"
PropertyTag: 0x66760003 (PidTagRuleSequence (section 2.2.1.3.1.2))
PropertyValue: 0x0000000A

PropertyTag: 0x66770003 (PidTagRuleState (section 2.2.1.3.1.3))
PropertyValue: 0x00000001 (ST_ENABLED)

PropertyTag: 0x667900FD (PidTagRuleCondition (section 2.2.1.3.1.9))
PropertyValue: "RES_CONTENT" condition, FuzzyLevel of 0x00010001 (FL_SUBSTRING | FL_IGNORECASE), where PropertyTag 0x003701F (PidTagSubject ([MS-OXPROPS] section 2.1034) contains "Project X". For more information, see section 2.2.1.

PropertyTag: 0x668000FE (PidTagRuleActions (section 2.2.1.3.1.10))
PropertyValue: 0x0001 actions (2), 0x00D0 bytes long, to ActionType is 0x01 ("OP_MOVE"), ActionFlavor 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.

PropertyTag: 0x6681001F (PidTagRuleProvider (section 2.2.1.3.1.5))
PropertyValue: Unicode string: "RuleOrganizer"
PropertyTag: 0x66830003 (PidTagRuleLevel (section 2.2.1.3.1.6))
PropertyValue: 0x00000000

PropertyTag: 0x66840102 (PidTagRuleProviderData (section 2.2.1.3.1.8))
PropertyValue: BLOB, 0x0010 bytes long, set by the client.

4.1.2 Server Responds to Client Request
A complete ROP response buffer in this example would appear as follows.

0000: 41 01 00 00 00 00

RopId: 0x41 (RopModifyRules ([MS-OXCROPS] section 2.2.11.1))
InputHandleIndex: 0x01
ReturnValue: 0x00000000. This response indicates the client has successfully created the rule (2).

4.2 Displaying Rules to the User
In this example, a client is required to display a list of active rules (2) on a folder to a user. The client sends a RopGetRulesTable ROP request ([MS-OXCROPS] section 2.2.11.2), using the buffer format specified in section 2.2.2. The client also sends RopSetColumns ([MS-OXCROPS] section 2.2.5.1) and RopQueryRows ROP requests ([MS-OXCROPS] section 2.2.5.4), using the buffer format described in [MS-OXCROPS] and [MS-OXCTABL].

4.2.1 Client Request for a Rules Table
A complete ROP request buffer to request a rules table would appear as follows.

0000: 3f 00 00 01 40

RopId: 0x3F (RopGetRulesTable ([MS-OXCROPS] section 2.2.11.2))
LogonId: 0x00
InputHandleIndex: 0x00
OutputHandleIndex: 0x01
TableFlags: 0x40 (specifying a Unicode table)

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

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

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

**RopId**: 0x12 (**RopSetColumns**)

**LogonId**: 0x00

**InputHandleIndex**: 0x01

**WantAsync**: 0x00 (Wait)

**PropertyTagCount**: 3

**PropertyTag1**: 0x66740014 (**PidTagRuleId** ([MS-OXPROPS] section 2.950))

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

**PropertyTag3**: 0x6682001f (**PidTagRuleName** ([MS-OXPROPS] section 2.960))

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

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

**RopId**: 0x15 (**RopQueryRows**)

**LogonId**: 0

**InputHandleIndex**: 1

**WantCurrentRow**: "FALSE" (Advance)

**WantForwardRead**: "TRUE" (forward reading)

**RowCount**: 50

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

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

**HandleIndex 0**: 0x00000223

**HandleIndex 1**: 0xffffffff

Note that the **HandleIndex[0]** field is referenced only in the **RopGetRulesTable** ROP request – it
refers to a table handle previously returned by the **RopOpenFolder** ROP ([MS-OXCROPS] section
2.2.4.1) (the Inbox, for example). The **HandleIndex[1]** field is referenced by the **RopGetRulesTable**
(as the new rules table index), the **RopSetColumns** (as the referenced table) and **RopQueryRows**
4.2.2 Server Responds to Client Requests

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

0000: 3f 01 00 00 00 00

RopId: 0x3F (RopGetRulesTable)
InputHandleIndex: 1
ReturnValue: 0x00000000. This response indicates the client has successfully gotten a handle to the rules table for the specified folder.

0000: 12 01 00 00 00 00

RopId: 0x12 (RopSetColumns)
InputHandleIndex: 1
ReturnValue: 0x00000000. This response indicates the client has successfully set the columns of the rules table.
CompletionStatus: 0x00 (TBLSTAT_COMPLETE ([MS-OXCTABL] section 2.2.2.1.3))

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

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

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

0000: 15 01 00 00 00 00 02 01 00

RopId: 0x15 (RopQueryRows)
InputHandleIndex: 1
ReturnValue: 0x00000000. This response indicates the RopQueryRows ROP call was successful.
Bookmark: 0x02 (BOOKMARK_END ([MS-OXCTABL] section 2.2.2.1.1))
RowCount: 1

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

Has Flag: "FALSE"

**Property 1**: 0x56F83F0100000001

**Property 2**: 0x10 byte binary array

**Property 3**: "Project X"

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

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

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

**Handle 0**: 0x00000223

**Handle 1**: 0x00000221

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

### 4.3 Deleting a Rule

In this example, a client is required to delete the **rule (2)** created in section 4.1 using the **RopModifyRules ROP** ([MS-OXCROPS] section 2.2.11.1). The client sends a **RopModifyRules ROP request**, using the buffer format described in section 2.2.1.

#### 4.3.1 Client Request Buffer

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

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

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

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

**RopId**: 0x41 (**RopModifyRules**)
LogonId: 0x00
InputHandleIndex: 0x00
ModifyRulesFlags: 0x00
RulesCount: 0x0001

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

0006: 04 01 00

RuleDataFlags: 0x04 (ROW_REMOVE)
PropertyValueCount: x0001

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

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

PropertyTag: 0x66740014 (PidTagRuleId (section 2.2.7.8))
PropertyValue: 0x56F83F0100000001

4.3.2 Server Responds to Client Request

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

0000: 41 00 00 00 00 00

RopId: 0x41 (RopModifyRules ([MS-OXCROPS] section 2.2.11.1))
InputHandleIndex: 0x00
ReturnValue: 0x00000000. This response indicates the client has successfully removed the rule (2).
5 Security

5.1 Security Considerations for Implementers

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

5.2 Index of Security Parameters

None.
6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Microsoft Exchange Server 2003
- Microsoft Exchange Server 2007
- Microsoft Exchange Server 2010
- Microsoft Exchange Server 2013
- Microsoft Exchange Server 2016
- Microsoft Exchange Server 2019
- Microsoft Office Outlook 2003
- Microsoft Office Outlook 2007
- Microsoft Outlook 2010
- Microsoft Outlook 2013
- Microsoft Outlook 2016
- Microsoft Outlook 2019
- Microsoft Outlook 2021
- Microsoft Outlook 2024 Preview

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

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

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

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

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

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

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

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

"JunkEmailRule", which is a rule that is created to help with Junk E-mail filtering
"MSFT:MR", which is a rule that assists the "Moderator" role on a public folder
"MSFT:Public.Folder.FormRestrictions", which is used by public folder rules

"ExchangeMailboxRules14", which is for rules that are specific to Exchange 2010, Exchange 2013, Exchange 2016, and Exchange 2019; rules with this provider string are not processed by Office Outlook 2003 or Office Outlook 2007.

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

Section 2.2.5.1: Office Outlook 2003, Office Outlook 2007, Outlook 2010, Outlook 2013, Outlook 2016, and Outlook 2019 set the ActionFlags field to 0x00000000.

Section 2.2.5.1.1: Exchange 2003 and Exchange 2007 do not support forwarding messages as SMS text messages.

Section 2.2.5.1.1: Exchange 2007, Exchange 2010, Exchange 2013, Exchange 2016, and Exchange 2019 send a reply message if the ActionType is "OP_OOF_REPLY".

Section 2.2.5.1.2.4.1: Exchange 2003 and Exchange 2007 also require the PidTagEntryId property for action "OP_FORWARD".

Section 2.2.5.1.2.4.1: Exchange 2003 and Exchange 2007 set this value to 0x01. Exchange 2010, Exchange 2013, Exchange 2016, and Exchange 2019 set this value to 0x00.

Section 2.2.5.1.2.7: Exchange 2003, Exchange 2007, Exchange 2010, Exchange 2013, Exchange 2016, and Exchange 2019 perform a hard delete, as described in [MS-OXCFOLD], but this is not required for the protocol.

Section 2.2.7.6: Exchange 2007 returns "ecNotFound".

Section 2.2.7.7: Exchange 2007 returns "ecNotFound".

Section 3.1.4.2.1: The server behavior is undefined in Microsoft Exchange Server 2013 Service Pack 1 (SP1).

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

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

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

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

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

Section 3.2.5.1: Microsoft Exchange Server 2007 Service Pack 3 (SP3) does not send a reply message.

Section 3.2.5.1: In Exchange 2007 SP3, the message appears in the user's mailbox.
<19> **Section 3.2.5.1.1:** Exchange 2013 SP1 evaluates the rule if the sender is on the list of recipients.

<20> **Section 3.2.5.2:** Exchange 2007, Exchange 2010, Exchange 2013, Exchange 2016, and Exchange 2019 limit the total aggregate size of standard rules (2) stored by a user to between 32 kilobytes (KB) and 256 KB; the exact limit is configured by the server administrator. Exchange 2007 and Exchange 2010 also enforce a limit of 500 disabled rules. Exchange 2013, Exchange 2016, and Exchange 2019 do not enforce a limit on disabled rules. Exchange 2003 limits the total aggregate size of all rules to 32 KB.

<21> **Section 3.2.5.2:** Exchange 2003, Exchange 2007, and Exchange 2010 do not save any of the changes when the changes push the rules (2) beyond the limit.

<22> **Section 3.2.5.3:** Exchange 2007 ignores the x bits and returns ecSuccess in this case.
7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

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

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

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

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Revision class</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 Appendix A: Product Behavior</td>
<td>Updated list of supported products.</td>
<td>Major</td>
</tr>
</tbody>
</table>
8 Index

A

Abstract data model
  client 34
  server 38

Abstract data model object types - client
  per deferred actions contents table 34

Abstract data model object types - server
  per mailbox 38
  per message 38
  per rule 38
  per rules table 38

Adding a new rule example
  client request buffer 45
  overview 45
  server responds to client request 48

Applicability 12

C

Capability negotiation 12
Change tracking 57

Client
  abstract data model 34
  initialization 34
  message processing 37
  other local events 38
  sequencing rules 37
  timer events 38
  timers 34

Client - abstract data model object types
  per deferred action contents table 34

Client - higher-layer triggered events
  adding rules 35
  deleting rules 35
  downloading a message to a different store 36
  modifying rules 35
  processing DAMs and DEMs 37
  retrieving existing rules 34

Client request buffer
  adding a new rule example 45
  deleting a rule example 51

Client request for a rules table
  displaying rules to the user example 48

Creating rules 11

D

DAM syntax
  PidTagClientActions property 30
  PidTagDamBackPatched property 29
  PidTagDamOriginalEntryId property 30
  PidTagDeferredActionMessageOriginalEntryId property 30
  PidTagMessageClass property 29
  PidTagRuleFolderEntryId property 30
  PidTagRuleId property 30
  PidTagRuleProvider property 30

DAM Syntax message 29

Data model - abstract
  client 34
  server 38

Deleting a rule example
  client request buffer 51
  overview 51
  server responds to client request 52

Deleting rules 11

DEM syntax
  PidTagDamOriginalEntryId property 32
  PidTagMessageClass property 31
  PidTagRuleActionNumber property 32
  PidTagRuleActionType property 31
  PidTagRuleError property 31
  PidTagRuleFolderEntryId property 32
  PidTagRuleId property 32
  PidTagRuleProvider property 32

DEM Syntax message 31

Displaying rules to the user example
  client request for a rules table 48
  overview 48
  server responds to client requests 50

E

Examples
  adding a new rule 45
  deleting a rule 51
  Displaying rules to the user 48
  overview 45

Executing client-side rules 12

Extended rules message syntax
  NamedPropertyInformation structure 21
  properties of an extended rule 18

Extended Rules Message Syntax message 18

F

Fields - vendor-extensible 12

G

Glossary 7

H

Higher-layer triggered events - client
  adding rules 35
  deleting rules 35
  downloading a message to a different store 36
  modifying rules 35
  retrieving existing rules 34

Higher-layer triggered events - server
  entering and exiting the Out of Office state 39
  processing incoming messages to a folder 39
  returning and maintaining the rules table 39

Higher-layer triggered events - client
  processing DAMs and DEMs 37

I

Implementer - security considerations 53

Index of security parameters 53
Informative references 10
Initialization
  client 34
  server 39
Introduction 7

M
Message processing
  client 37
  server 39
Message processing - server
  processing RopGetRulesTable 43
  processing RopModifyRules 42
  processing RopUpdateDeferredActionMessages 43
Message syntax 13
Messages
  DAM Syntax 29
  DEM Syntax 31
  Extended Rules Message Syntax 18
  RopGetRulesTable ROP 17
  RopModifyRules ROP 13
  RopUpdateDeferredActionMessages ROP 17
  RuleAction Structure 22
  syntax 13
  transport 13
Modifying rules 11

N
Normative references 10

O
Other local events
  client 38
  server 44
Overview
  executing client-side rules 12
  retrieving rules from the server 11
Overview - creating rules 11
Overview - deleting rules 11
Overview - modifying rules 11
Overview (synopsis) 10

P
Parameters - security index 53
Per deferred actions contents table abstract data model object type
  client 34
Per mailbox abstract data model object type
  server 38
Per message abstract data model object type
  server 38
Per rule abstract data model object type
  server 38
Per rules table abstract data model object type
  server 38
Preconditions 12
Prerequisites 12
Product behavior 54

R
References 10
  informative 10
  normative 10
Relationship to other protocols 12
Retrieving rules from the server 11
RopGetRulesTable ROP message 17
RopModifyRules format
  RuleData Structure 14
RopModifyRules ROP
  request buffer 13
  response buffer 14
RopModifyRules ROP message 13
RopUpdateDeferredActionMessages ROP
  request buffer 18
  response buffer 18
RopUpdateDeferredActionMessages ROP message 17
RuleAction structure
  ActionBlock structure 22
RuleAction Structure message 22
Rules-related message properties
  PidTagHasRules property 32
Rules-related message properties
  PidTagHasDeferredActionMessages property 33
  PidTagReplyTemplateId property 33
  PidTagRwRulesStream property 33

S
Security
  implementer considerations 53
  parameter index 53
Sequencing rules
  client 37
  server 39
Sequencing rules - server
  processing RopGetRulesTable 43
  processing RopModifyRules 42
  processing RopUpdateDeferredActionMessages 43
Server
  abstract data model 38
  initialization 39
  message processing 39
  other local events 44
  sequencing rules 39
  timer events 44
  timers 39
Server - abstract data model object types
  per mailbox 38
  per message 38
  per rule 38
  per rules table 38
Server - higher-layer triggered events
  processing incoming messages to a folder 39
Server - higher-layer triggered events
  entering and exiting the Out of Office state 39
  returning and maintaining the rules table 39
Server - message processing
  processing RopGetRulesTable 43
  processing RopModifyRules 42
  processing RopUpdateDeferredActionMessages 43
Server - sequencing rules
  processing RopGetRulesTable 43
  processing RopModifyRules 42
  processing RopUpdateDeferredActionMessages 43
Server responds to client request

- adding a new rule example 48
- deleting a rule example 52
- displaying rules to the user example 50
- Standards assignments 12

T

Timer events

- client 38
- server 44

Timers

- client 34
- server 39

Tracking changes 57

Transport 13

Triggered events - client

- adding rules 35
- deleting rules 35
- downloading a message to a different store 36
- modifying rules 35
- processing DAMs and DEMs 37
- retrieving existing rules 34

Triggered events - server

- entering and exiting the Out of Office state 39
- processing incoming messages to a folder 39
- returning and maintaining the rules table 39

V

Vendor-extensible fields 12

Versioning 12