[MS-OXOCFG]:
Configuration Information Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1	Major	Initial Availability.
04/25/2008	0.2	Minor	Revised and updated property names and other technical content.
06/27/2008	1.0	Major	Initial Release.
08/06/2008	1.0.1	Editorial	Revised and edited technical content.
09/03/2008	1.0.2	Editorial	Revised and edited technical content.
12/03/2008	1.0.3	Editorial	Revised and edited technical content.
04/10/2009	2.0	Major	Updated technical content for new product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	3.1.0	Minor	Updated the technical content.
02/10/2010	3.2.0	Minor	Updated the technical content.
05/05/2010	3.3.0	Minor	Updated the technical content.
08/04/2010	3.4	Minor	Clarified the meaning of the technical content.
11/03/2010	3.5	Minor	Clarified the meaning of the technical content.
03/18/2011	3.5	No change	No changes to the meaning, language, and formatting of the technical content.
08/05/2011	4.0	Major	Significantly changed the technical content.
10/07/2011	4.1	Minor	Clarified the meaning of the technical content.
01/20/2012	5.0	Major	Significantly changed the technical content.
04/27/2012	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	5.0	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
10/08/2012	6.0	Major	Significantly changed the technical content.
02/11/2013	6.1	Minor	Clarified the meaning of the technical content.
07/26/2013	7.0	Major	Significantly changed the technical content.
11/18/2013	8.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction 8
1.1 Glossary 8
1.2 References 9
1.2.1 Normative References 9
1.2.2 Informative References 10
1.3 Overview 11
1.4 Relationship to Other Protocols 11
1.5 Prerequisites/Preconditions 13
1.6 Applicability Statement 13
1.7 Versioning and Capability Negotiation 13
1.8 Vendor-Extensible Fields 13
1.9 Standards Assignments 13

2 Messages 14
2.1 Transport 14
2.2 Message Syntax 14
2.2.1 Namespaces 14
2.2.2 Configuration Data Properties 14
2.2.2.1 PidTagRoamingDatatypes Property 14
2.2.2.2 PidTagRoamingDictionary Property 15
2.2.2.3 PidTagRoamingXmlStream Property 15
2.2.3 XML Format 15
2.2.4 Binary Format 16
2.2.5 Configuration Data 16
2.2.5.1 Dictionaries 16
2.2.5.1.1 Calendar Options 18
2.2.5.2 XML Streams 20
2.2.5.2.1 Working Hours 20
2.2.5.2.2 Category List 24
2.2.5.2.3 Retention and Archive Settings 29
2.2.6 View Definitions 34
2.2.6.1 PidTagViewDescriptorBinary Property 35
2.2.6.1.1 ColumnPacket Structure 37
2.2.6.1.2 RestrictionPacket Structure 40
2.2.6.1.2.1 LogicalAndRestriction Structure 41
2.2.6.1.2.2 LogicalOrRestriction Structure 42
2.2.6.1.2.3 LogicalNotRestriction Structure 43
2.2.6.1.2.4 ContentRestriction Structure 43
2.2.6.1.2.5 PropertyRestriction Structure 44
2.2.6.1.2.6 ComparePropsRestriction Structure 46
2.2.6.1.2.7 BitmaskRestriction Structure 46
2.2.6.1.2.8 SizeRestriction Structure 47
2.2.6.1.2.9 ExistRestriction Structure 48
2.2.6.1.2.10 SubObjectRestriction Structure 48
2.2.6.1.2.11 CommentRestriction Structure 49
2.2.6.1.3 Valid Property Types 50
2.2.6.1.4 PropertyValue Structure 51
2.2.6.2 PidTagViewDescriptorStrings Property 52
2.2.7 Folder Flags 53
2.2.7.1 Sub-property 53

2.2.7.1.1 Invalid 54
2.2.7.1.2 ExtendedFlags 54
2.2.7.1.3 SearchFolderID 55
2.2.7.1.4 ToDoFolderVersion 55
2.2.8 Conversation Actions 56
2.2.8.1 PidLidConversationActionLastAppliedTime Property 56
2.2.8.2 PidLidConversationActionMaxDeliveryTime Property 56
2.2.8.3 PidLidConversationActionMoveFolderEid Property 57
2.2.8.4 PidLidConversationActionMoveStoreEid Property 57
2.2.8.5 PidLidConversationActionVersion Property 57
2.2.8.6 PidLidConversationProcessed Property 58
2.2.8.7 PidNameKeywords Property 58
2.2.8.8 PidTagConversationIndex Property 58
2.2.8.9 PidTagMessageClass Property 58
2.2.8.10 PidTagSubject Property 59
2.2.9 Navigation Shortcuts 59
2.2.9.1 PidTagMessageClass Property 59
2.2.9.2 PidTagNormalizedSubject Property 59
2.2.9.3 PidTagWlinkGroupHeaderID Property 59
2.2.9.4 PidTagWlinkSaveStamp Property 59
2.2.9.5 PidTagWlinkType Property 60
2.2.9.6 PidTagWlinkFlags Property 60
2.2.9.7 PidTagWlinkOrdinal Property 61
2.2.9.8 PidTagWlinkEntryId Property 61
2.2.9.9 PidTagWlinkRecordKey Property 61
2.2.9.10 PidTagWlinkStoreEntryId Property 61
2.2.9.11 PidTagWlinkFolderType Property 61
2.2.9.12 PidTagWlinkGroupClsid Property 62
2.2.9.13 PidTagWlinkGroupName Property 62
2.2.9.14 PidTagWlinkSection Property 62
2.2.9.15 PidTagWlinkCalendarColor Property 63
2.2.9.16 PidTagWlinkAddressBookEID Property 63
2.2.9.17 PidTagWlinkAddressBookStoreEID Property 63
2.2.9.18 PidTagWlinkClientID Property 64
2.2.9.19 PidTagWlinkROGroupType Property 64

3 Protocol Details 65
3.1 Client Details 65
3.1.1 Abstract Data Model 65
3.1.2 Timers 65
3.1.3 Initialization 65
3.1.3.1 Navigation Shortcuts 65
3.1.4 Higher-Layer Triggered Events 65
3.1.4.1 Reading Configuration Data 65
3.1.4.1.1 Reading Dictionaries 66
3.1.4.1.2 Reading Working Hours 67
3.1.4.1.3 Reading Category List 67
3.1.4.2 Writing Configuration Data 68
3.1.4.2.1 Writing Dictionaries 68
3.1.4.2.2 Writing Working Hours 68
3.1.4.2.3 Writing Category List 69
3.1.4.3 Reading View Definitions 69
3.1.4.4 Writing View Definitions 69

3.1.4.5 Reading Folder Flags 71
3.1.4.5.1 Reading ExtendedFolderFlags 71
3.1.4.5.2 Reading SearchFolderID 71
3.1.4.5.3 Reading ToDoFolderVersion 71
3.1.4.6 Writing Folder Flags 71
3.1.4.6.1 Writing ExtendedFolderFlags 71
3.1.4.6.2 Writing SearchFolderID 72
3.1.4.6.3 Writing ToDoFolderVersion 72
3.1.4.7 Applying a Category to a Message 72
3.1.4.8 Performing a Conversation Action 72
3.1.4.9 Reading Navigation Shortcuts 74
3.1.4.10 Writing Navigation Shortcuts 74
3.1.4.10.1 Group Headers 74
3.1.4.10.2 Shortcuts 75
3.1.5 Message Processing Events and Sequencing Rules 75
3.1.5.1 Processing a Conversation Action on Incoming E-mail Objects 75
3.1.5.1.1 Duplicate Detection for Conversation Action Processing 76
3.1.5.2 Processing a Conversation Action on Outgoing E-mail Objects 77
3.1.6 Timer Events 77
3.1.6.1 Expiration of Conversation Actions 77
3.1.7 Other Local Events 77
3.2 Server Details 77
3.2.1 Abstract Data Model 77
3.2.2 Timers 77
3.2.3 Initialization 78
3.2.4 Higher-Layer Triggered Events 78
3.2.4.1 Reading Configuration Data 78
3.2.4.1.1 Reading Working Hours 78
3.2.4.1.2 Reading Category List 78
3.2.4.2 Writing Configuration Data 78
3.2.4.3 Reading View Definitions 78
3.2.4.4 Reading Folder Flags 79
3.2.4.4.1 Reading ExtendedFolderFlags 79
3.2.4.4.2 Reading SearchFolderID 79
3.2.4.5 Writing Folder Flags 79
3.2.4.5.1 Writing ExtendedFolderFlags 79
3.2.4.5.2 Writing ToDoFolderVersion 79
3.2.4.6 Applying a Category to a Message 79
3.2.5 Message Processing Events and Sequencing Rules 79
3.2.5.1 Processing a Change to a Conversation Action FAI Message 79
3.2.5.2 Processing a Conversation Action on Incoming E-mail Objects 80
3.2.5.3 Processing a Conversation Action on Outgoing E-mail Objects 81
3.2.6 Timer Events 81
3.2.7 Other Local Events 81

4 Protocol Examples 82
4.1 Configuration Data 82
4.1.1 Dictionaries 82
4.1.2 Working Hours 82
4.1.3 Category List 83
4.2 View Definitions 84
4.2.1 PidTagViewDescriptorBinary 85
4.2.1.1 Blank Column 87

4.2.1.2 Column "Importance" 88
4.2.1.3 Column "Reminder" 89
4.2.1.4 Column "Icon" 90
4.2.1.5 Column "Flag Status" 90
4.2.1.6 Column "Attachment" 91
4.2.1.7 Column "From" 91
4.2.1.8 Column "Subject" 91
4.2.1.9 Column "Received" 92
4.2.1.10 Column "Size" 92
4.2.1.11 Column "Categories" 92
4.2.2 PidTagViewDescriptorStrings 93
4.3 Conversation Actions 93
4.3.1 A Categorized and Moved Conversation Action 93
4.4 Navigation Shortcut 95
4.4.1 Group Header 95
4.4.2 Navigation Shortcut 96

5 Security 98
5.1 Security Considerations for Implementers 98
5.2 Index of Security Parameters 98

6 Appendix A: Product Behavior 99

7 Change Tracking 103

8 Index 105

1 Introduction

The Configuration Information Protocol allows a client to share overlapping application settings with a server. Where appropriate, it can also be used to change the configuration of a feature on the client from the server or vice versa.

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

1.1 Glossary

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

code page
Coordinated Universal Time (UTC)
GUID
handle
little-endian
Unicode
UTF-8
XML
XML namespace

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

action
archive policy
attachments table
calendar
Calendar folder
Calendar object
calendar options dictionary
Calendar special folder
category
Common Views folder
Contact object
contents table
conversation
conversation action
conversation ID
dictionary
display name
distribution list
Email object
EntryID
FAI contents table
folder associated information (FAI)
free/busy status
group header
Inbox folder
Internet Message Access Protocol - Version 4 (IMAP4)
interpersonal messaging subtree
Journal object

mailbox
Meeting Request object
Meeting Response object
message class
Message object
message store
named property
Note object
permission
property ID
property tag
property type
public folder
recipient table
reminder
remote operation (ROP)
restriction
retention policy
Root folder
ROP request
ROP request buffer
rule
search folder
sort order
special folder
stream
Stream object
Task object
Web Distributed Authoring and Versioning Protocol (WebDAV)
working hours
XML namespace prefix
XML schema
XML schema definition (XSD)

The following terms are specific to this document:

subproperty: A binary stream property that is embedded in another property, possibly in addition to other subproperties.

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

1.2 References

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

1.2.1 Normative References

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

http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

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

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

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

[MS-OXCFXICS] Microsoft Corporation, "Bulk Data Transfer Protocol".

[MS-OXCICAL] Microsoft Corporation, "iCalendar to Appointment Object Conversion Algorithm".

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

[MS-OXCPERM] Microsoft Corporation, "Exchange Access and Operation Permissions Protocol".

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

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

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

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

[MS-OXOCAL] Microsoft Corporation, "Appointment and Meeting Object Protocol".

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

[MS-OXORMDR] Microsoft Corporation, "Reminder Settings Protocol".

[MS-OXORULE] Microsoft Corporation, "Email Rules Protocol".

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

[MS-OXOSRCH] Microsoft Corporation, "Search Folder List Configuration Protocol".

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

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

[XMLBase] Marsh, J., and Tobin, R., Eds., "XML Base (Second Edition)", W3C Recommendation, January 2009, http://www.w3.org/TR/2009/REC-xmlbase-20090128/

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA2] Biron, P.V., and Malhotra, A., Eds., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

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

[MS-OXCMAPIHTTP] Microsoft Corporation, "Messaging Application Programming Interface (MAPI) Extensions for HTTP".

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

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

[MS-OXOFLAG] Microsoft Corporation, "Informational Flagging Protocol".

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

1.3 Overview

The Configuration Information Protocol consists of the settings that are shared between clients and servers, and the manner in which the settings are shared. The protocol enables both clients and servers to implement features based on user configuration information, such as using a user's preferred **working hours**, to propose optimal times for new appointments. The settings are divided into the following categories, each of which uses a different mechanism for sharing:

- configuration data
- view definitions
- folder flags
- conversation actions
- navigation shortcuts

In addition to the settings included in the protocol, a client or server can store additional, non-interoperable settings for the use of the respective application in a similar way. Despite the fact that the settings use a similar storage mechanism, they are not part of the protocol when they are used by only a single application.

1.4 Relationship to Other Protocols

The Configuration Information Protocol works with the protocols shown in the following diagram.

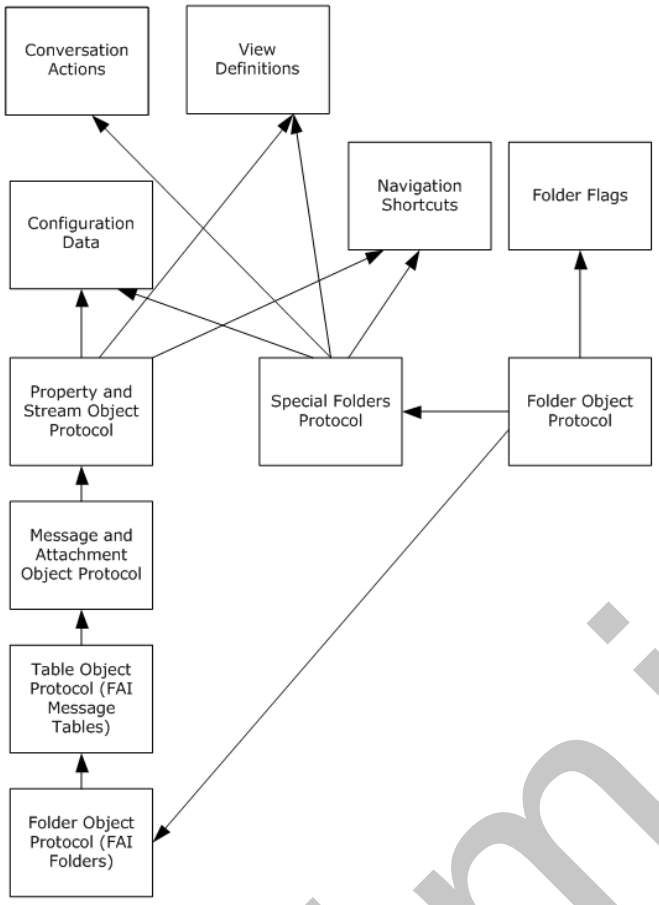


Figure 1: Components of this protocol in relation to other protocols

The configuration data, view definition, and navigation shortcut components of the Configuration Information Protocol use **folder associated information (FAI)** messages, as described in [MS-OXCFOLD], as the transport. Clients need to sort and restrict the **FAI contents table**, as described in [MS-OXCTABL], to find the required FAI message. Clients need to access properties on the FAI message, as described in [MS-OXCMSG], and need to use **Stream objects** for certain properties, as described in [MS-OXCPRPT]. The FAI messages are sometimes contained in a **special folder**; therefore, these components need to use the Special Folders Protocol, as described in [MS-OXOSFLD]. The configuration data and view definitions are retrieved from properties on the FAI messages using the Property and Stream Object Protocol, as described in [MS-OXCPRPT]. Conversation actions are stored as FAI messages in the conversation actions settings special folder, as described in [MS-OXOSFLD].

The folder flags component uses a binary property that is stored on the folder itself. The transport for folder flags is described in [MS-OXCFOLD].

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

1.5 Prerequisites/Preconditions

The Configuration Information Protocol assumes that the client has previously logged on to the server.

1.6 Applicability Statement

Clients and servers can use the Configuration Information Protocol to share application settings when each application implements a similar feature with the same settings. Each application can also use this protocol to communicate the state of its own features, where that state affects the state of related features in the other application.

Clients can also use this protocol to synchronize application settings between multiple instances of the client that are connected to the same server.

1.7 Versioning and Capability Negotiation

Capability negotiation: The client checks the version number returned by the server in either the **EcDoConnectEx** method, as described in [MS-OXCRPC], or the **X-ServerApplication** header of the **Connect** request type response,<1> as described in [MS-OXCMAPIHTTP]. If the server returns a version that is greater than or equal to 14.0.324.0, the client never deletes the conversation action FAI message, as described in section 3.1.4.8, and disables almost all processing of incoming **E-mail objects**, as described in section 3.1.5.1.

1.8 Vendor-Extensible Fields

A third-party application can use FAI messages to store its own settings by specifying its own custom **PtypString** for the value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3). A centralized authority that ensures uniqueness of the **PidTagMessageClass** property across different applications does not exist.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

Section 2.2 specifies how Configuration Information Protocol messages use properties and **streams (2)** that have been set on FAI messages or folders as the underlying transport.

Streams (2) are further specified in [MS-OXCPRPT]. FAI messages and folders are further specified in [MS-OXCFOLD].

2.2 Message Syntax

Sections 2.2.1 through 2.2.9.19 specify the location and format of the property and stream (2) buffers that are specific to the Configuration Information Protocol.

2.2.1 Namespaces

This specification defines and references various **XML namespaces** using the mechanisms specified in [XMLNS]. Although this specification associates a specific **XML namespace prefix** for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

Prefix	Namespace URI	Reference
None	dictionary	
None	WorkingHours	
None	CategoryList	

2.2.2 Configuration Data Properties

Configuration data consists of groups of related application settings. Each group of settings is stored in separate stream (2) properties that are set on FAI messages.

The streams (2) can contain a serialized **dictionary** of name-value pairs that allow access to individual settings by name. The dictionary is serialized using an **XML schema** that is common to all dictionary streams (2). Most simple settings use this type of stream (2).

For more structured data, such as the user's preferred working hours, the streams (2) can contain an **XML** document that uses an arbitrary schema that corresponds to the structure of the data. The settings that use an arbitrary XML stream (2) include the user's preferred working hours, which can be used by the client and server to make improved scheduling suggestions for that user, and the user's customized category list, which allows the user to build a list of commonly used message **categories (3)** and assign color values to those categories (3).

The properties specified in sections 2.2.2.1 through 2.2.2.3 are present on FAI messages that contain configuration data.

2.2.2.1 PidTagRoamingDatatypes Property

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

The **PidTagRoamingDatatypes** property ([MS-OXPROPS] section 2.925) contains a bitmask that indicates which stream (2) properties exist on the message. The types of the streams (2), and thus the flags, are not mutually exclusive.

The bitmask is a bitwise OR of of the following bits. Other bits MUST NOT be set and MUST be ignored if set.

Value	Meaning
0x00000002	The FAI message contains an XML stream (2), which is stored in the **PidTagRoamingXmlStream** property (section 2.2.2.3).
0x00000004	The FAI message contains a dictionary stream (2), which is stored in the **PidTagRoamingDictionary** property (section 2.2.2.2). If the FAI message does not contain a dictionary stream (2), the application MUST treat the dictionary as having no entries.

2.2.2.2 PidTagRoamingDictionary Property

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

The **PidTagRoamingDictionary** property ([MS-OXPROPS] section 2.926) contains a dictionary stream (2) that is serialized into a fixed XML schema, as specified in section 2.2.5.1.

2.2.2.3 PidTagRoamingXmlStream Property

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

The **PidTagRoamingXmlStream** property ([MS-OXPROPS] section 2.927) contains an XML stream (2) that uses an arbitrary XML schema, as specified in section 2.2.5.2.

2.2.3 XML Format

The supported XML format to be read and written as configuration data, as specified in section 2.2.2, is a subset of the W3C recommendation specified in [XMLBase].

Applications MUST NOT depend on support for namespaces. Namespace support is specified in [XMLBase].

Applications MUST NOT output XML with namespaces except to declare the default namespace if specified in this protocol.

Applications MUST remove namespace prefixes from any qualified name in the default namespace.

Applications MUST escape the following special characters within quoted strings:

Special character	Escape sequence
"	&quot;
<	&lt;
>	&gt;
&	&amp;

Applications SHOULD<2> escape the following special characters within quoted strings:

Special character	Escape sequence
'	&apos;

2.2.4 Binary Format

Unless otherwise specified, the application serializes multibyte data types into binary streams (2) using the **little-endian** byte order.

2.2.5 Configuration Data

The client and server SHOULD<3> store certain settings as configuration data, as specified in section 2.2.2. The format and location of the configuration data, as well as which settings this data can include, are specified in sections 2.2.5.1 through 2.2.5.2.3.

The application stores configuration data in an FAI message. The application stores the FAI message in the special folder that is specified in in the subsections of this section for each type of configuration data.

The message MUST have the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3 set. The value of the property MUST use the prefix "IPM.Configuration." followed by a name that uniquely identifies this FAI message in that folder.

The message MUST have the **PidTagRoamingDatatypes** property set as specified in section 2.2.2.1.

2.2.5.1 Dictionaries

A message with a dictionary stream (2) MUST have the **PidTagRoamingDatatypes** property (section 2.2.2.1) set. The value of the property MUST be a bitmask that includes the value 0x00000004.

The message MUST have the **PidTagRoamingDictionary** property (section 2.2.2.2) set. The value of the property is a binary stream (2) that contains a **Unicode** XML document using **UTF-8** encoding. The XML document MUST conform to the following **XML schema definition (XSD)** in addition to the limitations specified in section 2.2.3.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="dictionary.xsd"
           xmlns="dictionary.xsd"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="UserConfiguration">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Info">
          <xs:complexType>
            <xs:sequence />
            <xs:attribute name="version"
                          type="VersionString">
            </xs:attribute>
          </xs:complexType>
        </xs:element>
        <xs:element name="Data">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="e"
```

```
                        minOccurs="0"
                        maxOccurs="unbounded"
                        type="EntryType">
              </xs:element>
            </xs:sequence>
          </xs:complexType>
          <xs:unique name="uniqueKey">
            <xs:selector xpath="e" />
            <xs:field xpath="@k" />
          </xs:unique>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="VersionString">
    <xs:restriction base="xs:string">
      <xs:pattern value=".+\.\d+" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="EntryType">
    <xs:sequence />
    <xs:attribute name="k"
                  type="ValueString" />
    <xs:attribute name="v"
                  type="ValueString" />
  </xs:complexType>
  <xs:simpleType name="ValueString">


    <xs:restriction>
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="\d+\-.*" />
        </xs:restriction>
      </xs:simpleType>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

UserConfiguration: A top-level element that MUST exist. It contains the **Info** and **Data** elements.

Info: An element that MUST exist. It contains information about the application that created the XML document in the **version** attribute.

version: An attribute on the **Info** element that specifies the name and version of the application that created the XML document. The type of this attribute MUST be **VersionString**.

VersionString: A **simpleType** based on a string. The name and version of the application that created this document SHOULD<4> be encoded in the version string. The data is not validated; it is provided for future reference. The format of the version string is as follows.

```
<name>.<major version number>
```

Data: An element that MUST contain all the dictionary name-value pair entries.

e: An element of type **EntryType** that contains a name-value pair. There can be an unbounded number of **e** elements inside the top-level **Data** element.

EntryType: A **complexType** that represents a name-value pair. This type contains the following attributes:

- **k**: An attribute of the **EntryType** type that contains the name portion of the name-value pair. The type of this attribute is **ValueString**. The value of this attribute MUST be unique within the dictionary.
- **v**: An attribute of the **EntryType** type that contains the value portion of the name-value pair. The type of this attribute is **ValueString**.

ValueString: A **simpleType** that is based on a string. Different value types MUST be encoded in this **simpleType** as a string. The format of the string MUST be as follows.

```
<data type>-<string encoded value>
```

The data type MUST be an integer type code from the following table.

Type	Type code	Encoding
Boolean	3	"True" or "False"
32-bit signed integer	9	Decimal characters, prefixed with an optional "-" to denote a negative number
String	18	Unicode string

There is one reserved name-value pair that the client SHOULD include in every dictionary XML document. If the dictionary XML document does not include this name-value pair, the client MUST behave as though the default value were set:

- OLPrefsVersion
  - Name: (string) "OLPrefsVersion"
  - Value: (32-bit integer) The client uses this setting to determine whether to prefer the settings in the XML document or its own locally stored settings.
    - "0" or any negative integer: The client MUST prefer its own default or locally stored settings, and it MUST rewrite the XML document with those settings.
    - "1" or any positive integer: The client MUST prefer the settings in the XML document.
    - Default: (32-bit integer) "0"

2.2.5.1.1 Calendar Options

If the client or server supports configuration data as specified in section 2.2.2, it stores the settings specified in this section in a **calendar options dictionary**. The application stores the calendar options dictionary in an FAI message that is contained in the **Calendar special folder**. The format of the Calendar special folder is specified in [MS-OXOSFLD] section 2.2.8.

This message MUST have the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) set. The value of the property MUST be "IPM.Configuration.Calendar".

The dictionary SHOULD include the following settings.

Note Unless otherwise specified, any setting that is not included in the dictionary MUST revert to the default value.

- piRemindDefault
  - Name: (string) "piRemindDefault"
  - Value: (32-bit integer) When creating a new appointment, the client or server SHOULD initialize the **reminder** time to be the start time of the appointment minus this number of minutes, as specified in [MS-OXORMDR].
  - Default: (32-bit integer) "15"
- piReminderUpgradeTime
  - Name: (string) "piReminderUpgradeTime"
  - Value: (32-bit integer) The value of this setting is specified in [MS-OXORMDR] section 2.2.3.1.
  - Default: (missing) The default behavior when this setting is missing is specified in [MS-OXORMDR] section 3.1.3.1.
- piAutoProcess
  - Name: (string) "piAutoProcess"
  - Value: (Boolean) The client SHOULD use this setting to control automatic processing of **Meeting Request objects** and **Meeting Response objects**, as specified in [MS-OXOCAL].
    - "True": The client SHOULD enable automatic processing.
    - "False": The client SHOULD disable automatic processing.
    - Default: (Boolean) "True"
- AutomateProcessing
  - Name: (string) "AutomateProcessing"
  - Value: (32-bit integer) The server uses this setting to control automatic processing of Meeting Request objects and Meeting Response objects if it implements this feature. If the server does not implement this feature, the server MUST ignore this setting. This setting has three possible values:
    - "0": The server MUST disable automatic processing.
    - "1": The server MUST enable automatic processing, if it implements this feature.
    - "2": The server MUST enable automatic processing, if it implements this feature, treating the **Calendar object** as a meeting resource rather than an attendee, as specified in [MS-OXOCAL]. The client MUST NOT change the setting when it has this value.
    - Default: (32-bit integer) "1"

- piAutoDeleteReceipts
  - Name: (string) "piAutoDeleteReceipts"
  - Value: (Boolean) The client SHOULD<5> use this setting to control automatic deletion of Meeting Response objects, as specified in [MS-OXOCAL].
    - "True": The client enables automatic deletion.
    - "False": The client disables automatic deletion.
    - Default: (Boolean) "False"

2.2.5.2 XML Streams

The message MUST have the **PidTagRoamingDatatypes** property (section 2.2.2.1) set. The value of the property MUST be a bitmask that includes 0x00000002.

The message MUST have the **PidTagRoamingXmlStream** property (section 2.2.2.3) set. The value of the property MUST be a **PtypBinary** stream (2) that contains a Unicode XML document that is using the UTF8 encoding.

In addition to the XSDs that are specified in sections 2.2.5.2.1 through 2.2.5.2.3, the XML document MUST conform to the limitations specified in section 2.2.3.

If the application encounters unknown XML elements while parsing the document, it SHOULD preserve those elements without modification and include them whenever it makes modifications to the parts of the document that it understands.

2.2.5.2.1 Working Hours

If the client or server supports configuration data, as specified in section 2.2.2, it stores the settings that are specified in this section in a working hours stream (2). The application stores the working hours stream (2) in an FAI message contained in the Calendar special folder.

The message MUST have the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) set. The value of the property MUST be "IPM.Configuration.WorkHours".

The XML document that is stored in the **PidTagRoamingXmlStream** property (section 2.2.2.3) MUST conform to the following XSD.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="WorkingHours.xsd"
           xmlns="WorkingHours.xsd"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Root">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="WorkHoursVersion1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="TimeZone">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Bias"
                                type="xs:short"/>
                    <xs:element name="Standard"
```

```
                        type="DSTTransition" />
                    <xs:element name="DaylightSavings"
                        type="DSTTransition" />
                    <xs:element name="Name"
                        minOccurs="0"
                        maxOccurs="1"
                        type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="TimeSlot">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Start"
                        type="xs:time" />
                    <xs:element name="End"
                        type="xs:time" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="WorkDays"
                        type="WorkDaysList" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="DSTTransition">
    <xs:sequence>
      <xs:element name="Bias"
                type="xs:short" />
      <xs:element name="ChangeDate">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Time"
                      type="xs:time" />
            <xs:element name="Date">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:annotation>
                    <xs:documentation xml:lang="en-us">
                      The Date element is a date formatted as
                      "yyyy/mm/dd," where "yyyy" is the 4 digit
                      year, "mm" is the 2 digit month, and "dd"
                      is the 2 digit day of the month.
                    </xs:documentation>
                  </xs:annotation>
                  <xs:pattern value="\d{4}/\d{2}/\d{2}"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
            <xs:element name="DayOfWeek">
              <xs:simpleType>
                <xs:restriction base="xs:unsignedByte">
                  <xs:minInclusive value="0"/>
                  <xs:maxInclusive value="7"/>
                </xs:restriction>
              </xs:simpleType>
```

```
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="WorkDaysList">
    <xs:list itemType="WorkDayType"/>
  </xs:simpleType>
  <xs:simpleType name="WorkDayType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Monday"/>
      <xs:enumeration value="Tuesday"/>
      <xs:enumeration value="Wednesday"/>
      <xs:enumeration value="Thursday"/>
      <xs:enumeration value="Friday"/>
      <xs:enumeration value="Saturday"/>
      <xs:enumeration value="Sunday"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Root: The top-level element in the XML document. This element MUST exist. The application specifies the XML namespace on this element as "WorkingHours.XSD". This element MUST contain the **WorkHoursVersion1** element.

WorkHoursVersion1: This element MUST exist and contains the **TimeZone**, **TimeSlot**, and **WorkDays** elements.

TimeZone: This element MUST exist and contains a description of the user's current time-zone settings. It contains the **Bias**, **Standard**, **DaylightSavings**, and **Name** elements.

Bias: This element MUST exist as a subelement of the **TimeZone** element. It contains the offset in minutes of the user's current time zone from **Coordinated Universal Time (UTC)**.

Standard: This element MUST exist and contains the definition of standard time in the user's time zone. The type of this element is **DSTTransition**.

DaylightSavings: This element MUST exist and contains the definition of daylight-saving time in the user's time zone. The type of this element is **DSTTransition**.

Name: This element SHOULD<6> exist and contains the standard name of the time zone described by the data in the **TimeZone** element. For possible values, see the **KeyName** field of the **PidLidAppointmentTimeZoneDefinitionRecur** property ([MS-OXOCAL] section 2.2.1.41).

DSTTransition: This **complexType** describes the differences between standard time and daylight-saving time in the user's current time zone. It contains the **Bias** and **ChangeDate** elements. The Bias element from the **DSTTransition** type MUST be added to the **Bias** element value contained in the **WorkHoursVersion1** element when this transition takes effect, which MUST be determined by the value of the **ChangeDate** element.

Bias: This element MUST exist as a subelement of the **DSTTransition** type. The **Bias** element value specified in the **DSTTransition** type MUST be added to the time zone bias after the transition.

ChangeDate: This element MUST exist as a subelement of the **DSTTransition** type. The value of this element determines when the transition takes place. This element contains a **Time** element, a **Date** element, and a **DayOfWeek** element.

Time: This element contains the time of day when the transition takes place.

Date: This element contains a date formatted as <yyyy/mm/dd>, where yyyy is the 4-digit year, mm is the 2-digit month, and dd is the 2-digit day of the month.

If the year is set to "0000", the application performs the transition every year. If the year is any other value, the application performs the transition only in that year.

The application performs the transition in the month that is specified.

If the year is set to "0000", the interpretation of the day of the month depends on the value of the **DayOfWeek** element, as specified in this section. If the year is any other value, the application performs the transition on the day of the month that is specified.

DayOfWeek: If the year portion of the **Date** element is set to "0000", this element MUST contain the day of the week when the transition takes place. The application selects the occurrence of that day of the week using the day of the month portion of the **Date** element. For example, if the **DayOfWeek** element contains the value 0, and the day of the month is 2 in the **Date** element, the application performs the transition on the second Sunday of the month. In this case, the day of the month in the **Date** element MUST be between 1 and 5, inclusive. The possible values for the **DayOfWeek** element are given in the following table.

Value	Meaning
0	The application performs the transition on a Sunday.
1	The application performs the transition on a Monday.
2	The application performs the transition on a Tuesday.
3	The application performs the transition on a Wednesday.
4	The application performs the transition on a Thursday.
5	The application performs the transition on a Friday.
6	The application performs the transition on a Saturday.

If the year portion of the **Date** element is any other value, the application MUST ignore the **DayOfWeek** element and use the day of the month portion of the **Date** element instead.

TimeSlot: This element contains the **Start** and **End** elements.

Start: This element contains the start time for the user's work day, relative to the user's current time zone, as specified in the **TimeZone** element.

End: This element contains the end time for the user's work day, relative to the user's current time zone, as specified in the **TimeZone** element.

WorkDays: This element contains a list of strings that specify which days of the week are work days for this user. The set of strings is defined by the enumeration **restriction (1)** on the **WorkDayType simpleType**. The application treats any day that is included in this element as a work day for the user.

WorkDayType: A **simpleType** based on a string. The possible values are given in the following table.

Value	Meaning
Monday	If this string is included in the **WorkDays** element list, Monday is a work day for this user.
Tuesday	If this string is included in the **WorkDays** element list, Tuesday is a work day for this user.
Wednesday	If this string is included in the **WorkDays** element list, Wednesday is a work day for this user.
Thursday	If this string is included in the **WorkDays** element list, Thursday is a work day for this user.
Friday	If this string is included in the **WorkDays** element list, Friday is a work day for this user.
Saturday	If this string is included in the **WorkDays** element list, Saturday is a work day for this user.
Sunday	If this string is included in the **WorkDays** element list, Sunday is a work day for this user.

2.2.5.2.2 Category List

A category list is a type of configuration data, as specified in section 2.2.5, that contains a list of textual labels (or categories (3)) with associated data, such as color. Other attributes of a category (3) include a shortcut key that can be used to apply a category (3), a usage counter, the last time the category (3) was applied or used by the user, and a **GUID**.

If the client or server supports configuration data, as specified in section 2.2.2, it MUST store the settings that are specified in this section in a category list stream (2). The application MUST store the category list stream (2) in an FAI message that is contained in the Calendar special folder.

The message MUST have the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) set. The value of the property MUST be "IPM.Configuration.CategoryList".

The XML document that is stored in the **PidTagRoamingXmlStream** property (section 2.2.2.3) MUST conform to the following XSD.

```
<?xml version="1.0"?>
<xs:schema targetNamespace="CategoryList.xsd"
           xmlns="CategoryList.xsd"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="colorType">
    <xs:restriction base="xs:int">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="keyboardShortcutType">
    <xs:restriction base="xs:unsignedInt">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="dateTimeRestrictedType">
    <xs:restriction base="xs:dateTime">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="renameOnFirstUseType">
    <xs:restriction base="xs:int">
      <xs:enumeration value="0"/>
      <xs:enumeration value="1"/>
    </xs:restriction>
```

```
  </xs:simpleType>

  <xs:simpleType name="guidType">
    <xs:restriction base="xs:string">
      <xs:pattern value="^\{[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-
[0-9a-fA-F]{12}\}$"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="categories">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded"
                    name="category">
          <xs:complexType>
            <xs:attribute name="name"
                          type="xs:string"
                          use="required" />
            <xs:attribute name="color"
                          type="colorType"
                          use="required" />
            <xs:attribute name="keyboardShortcut"
                          type="keyboardShortcutType"
                          use="required" />
            <xs:attribute name="usageCount"
                          type="xs:unsignedInt"
                          use="optional" />
            <xs:attribute name="lastTimeUsedNotes"
                          type="dateTimeRestrictedType"
                          use="optional" />
            <xs:attribute name="lastTimeUsedJournal"
                          type="dateTimeRestrictedType"
                          use="optional" />
            <xs:attribute name="lastTimeUsedContacts"
                          type="dateTimeRestrictedType"
                          use="optional" />
            <xs:attribute name="lastTimeUsedTasks"
                          type="dateTimeRestrictedType"
                          use="optional" />
            <xs:attribute name="lastTimeUsedCalendar"
                          type="dateTimeRestrictedType"
                          use="optional" />
            <xs:attribute name="lastTimeUsedMail"
                          type="dateTimeRestrictedType"
                          use="optional" />
            <xs:attribute name="lastTimeUsed"
                          type="dateTimeRestrictedType"
                          use="required" />
            <xs:attribute name="lastSessionUsed"
                          type="xs:int"
                          use="required" />
            <xs:attribute name="guid"
                          type="guidType"
                          use="required" />
            <xs:attribute name="renameOnFirstUse"
                          type="renameOnFirstUseType"
                          use="optional" />
          </xs:complexType>
        </xs:element>
```

```
      </xs:sequence>
      <xs:attribute name="default"
                    type="xs:string"
                    use="required" />
      <xs:attribute name="lastSavedSession"
                    type="xs:int"
                    use="required" />
      <xs:attribute name="lastSavedTime"
                    type="dateTimeRestrictedType"
                    use="required" />
    </xs:complexType>
    <xs:unique name="uniqueName">
      <xs:selector xpath="category" />
      <xs:field xpath="@name" />
    </xs:unique>
  </xs:element>
</xs:schema>
```

colorType: A **simpleType** based on an integer. For possible values, see the description of the **color** attribute later in this section.

keyboardShortcutType: A **simpleType** based on an unsigned integer. For possible values, see the description of the **keyboardShortcut** attribute later in this section.

dateTimeRestrictedType: A **simpleType** based on a **dateTime** type as specified in [XMLSCHEMA2] section 3.2.7. Elements with this type have the following restrictions:

- The year MUST be between 1601 and 30827.
- The time 24:00:00 is not valid.
- Fractional seconds SHOULD have 3-digit precision (that is, milliseconds). The application can include additional digits. The application SHOULD<7> handle any extra digits if they are included.
- The application MUST specify the time in **UTC**. The application MAY append a Z for the time zone identifier. The application MUST ignore any other time zone identifier and interpret the time using UTC.

renameOnFirstUseType: A **simpleType** based on an integer. For possible values, see the description of the **renameOnFirstUse** attribute later in this section.

guidType: A **simpleType** based on a string. For possible values, see the description of the **guid** attribute later in this section.

categories: The top-level element in the XML document. This element MUST exist. The application specifies the XML namespace on this element as "CategoryList.xsd". This element MUST contain the **category** element.

category: This element MUST exist and contains the **name**, **color**, **keyboardShortcut**, **usageCount**, **lastTimeUsedNotes**, **lastTimeUsedJournal**, **lastTimeUsedContacts**, **lastTimeUsedTasks**, **lastTimeUsedCalendar**, **lastTimeUsedMail**, **lastTimeUsed**, **lastSessionUsed**, **guid**, and **renameOnFirstUse** attributes.

name: An attribute on the **category** element that describes the name of the category (3). A valid category name:

- MUST be unique in the category list (case insensitive).

- MUST NOT be empty.
- MUST NOT be longer than 255 characters.
- MUST NOT contain the comma character (,).
- SHOULD NOT contain the characters (;) (\x061B) (\xFE54) (\xFF1B).

It is also recommend that a category name not be in the form of the string representation of a GUID, as specified in the description of the **guid** attribute in this section.<8>

color: An attribute on the **category** element that describes the color of the category (3). The application SHOULD use a value from -1 to 24. If any other value is used, the application MUST interpret that value as if it were -1 (no color). The RGB values provided here are the basic colors for the category (3). Applications can choose to display the color category (3) differently.

Value	Base R,G,B	Color name
-1	255, 255,255	No color
0	214, 37, 46	Red
1	240, 108, 21	Orange
2	255, 202, 76	Peach
3	255, 254, 61	Yellow
4	74, 182, 63	Green
5	64, 189, 149	Teal
6	133, 154, 82	Olive
7	50, 103, 184	Blue
8	97, 61, 180	Purple
9	163, 78, 120	Maroon
10	196, 204, 221	Steel
11	140, 156, 189	Dark steel
12	196, 196, 196	Gray
13	165, 165, 165	Dark gray
14	28, 28, 28	Black
15	175, 30, 37	Dark red
16	177, 79, 13	Dark orange
17	171, 123, 5	Dark peach
18	153, 148, 0	Dark yellow
19	53, 121, 43	Dark green

Value	Base R,G,B	Color name
20	46, 125, 100	Dark teal
21	95, 108, 58	Dark olive
22	42, 81, 145	Dark blue
23	80, 50, 143	Dark purple
24	130, 55, 95	Dark maroon

keyboardShortcut: An attribute on the **category** element that describes the keyboard shortcut of the category (3).<9> The application SHOULD use a value from 0 to 11. If any other value is used, the application MUST interpret that value as if it were 0 (no shortcut).

Value	Shortcut key
0	None
1	CTRL+F2
2	CTRL+F3
3	CTRL+F4
4	CTRL+F5
5	CTRL+F6
6	CTRL+F7
7	CTRL+F8
8	CTRL+F9
9	CTRL+F10
10	CTRL+F11
11	CTRL+F12

usageCount: An attribute on the **category** element. Reserved. Applications can write 0. Applications MAY<10> write the usage count and periodically apply an algorithm to reduce the usage count to facilitate creating a most frequently used list.

lastTimeUsedNotes (optional): An attribute on the **category** element. This attribute contains the last time the category (3) was applied to a **Note object**.

lastTimeUsedJournal (optional): An attribute on the **category** element. This attribute contains the last time the category (3) was applied to a **Journal object**.

lastTimeUsedContacts (optional): An attribute on the **category** element. This attribute contains the last time the category (3) was applied to a **Contact object**.

lastTimeUsedTasks (optional): An attribute on the **category** element. This attribute contains the last time the category (3) was applied to a **Task object**.

lastTimeUsedCalendar (optional): An attribute on the **category** element. This attribute contains the last time the category (3) was applied to a Calendar object.

lastTimeUsedMail (optional): An attribute on the **category** element. This attribute contains the last time the category (3) was applied to an E-mail object.

lastTimeUsed: An attribute on the **category** element. This attribute contains the last time the category (3) was used. See section 3.1.4.2.3 for more details.

lastSessionUsed: An attribute on the **category**. Reserved. Applications SHOULD<11> write 0, but can write the last session that this category (3) was applied or changed by the user.

guid: An attribute on the **category** element that describes a GUID that identifies the category (3) and does not change if the user renames the category (3). The GUID MUST be stored in a string in the form of "{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}", where X is any hexadecimal digit.

renameOnFirstUse: An attribute on the **category** element. If set to "1", an application SHOULD<12> prompt the user to rename the category (3) when it is first applied to a message, as specified in section 3.1.4.7. If the user renames the category (3) before applying it to a message, this attribute can be set to 0. If this attribute is missing, the application MUST use a default value of 0.

default: An attribute on the **categories** element that specifies the name of a category (3) in the category list that is to be applied, as specified in section 3.1.4.7, if the application provides a one-click method to apply a category (3).

lastSavedSession: An attribute on the **categories** element. Reserved. Applications SHOULD write 0<13> but can choose to increment this number at every new session.

lastSavedTime: An attribute on the **categories** element. The value MUST be set to the time in UTC when the category list was saved.

2.2.5.2.3 Retention and Archive Settings

If the client or server supports configuration data, as specified in section 2.2.2, and the **retention policy** or **archive policy** features are enabled, then it SHOULD store the settings that are specified in this section in a tags list stream (2). A tags list is a type of configuration data, as specified in section 2.2.5, that contains a list of settings for a retention policy or an archive policy. The application SHOULD<14> store the tags list stream (2) in an FAI message that is contained in the **Inbox folder**. The message MUST have the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) set. The value of the property MUST be "IPM.Configuration.MRM". The XML document that is stored in the **PidTagRoamingXmlStream** property (section 2.2.2.3) MUST conform to the following XSD.

```
<?xml version="1.0"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="UserConfiguration">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Info">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Data">
                <xs:complexType>
                  <xs:sequence>
```

```
                    <xs:element name="RetentionHold">
                      <xs:complexType>
                        <xs:attribute name="Enabled" type="xs:string" use="required" />
                        <xs:attribute name="RetentionComment" type="xs:string" />
                        <xs:attribute name="RetentionUrl" type="xs:string" />
                      </xs:complexType>
                    </xs:element>
                    <xs:element maxOccurs="unbounded" name="PolicyTag">
                      <xs:complexType>
                        <xs:sequence minOccurs="0">
                          <xs:element minOccurs="0" name="LlocalizedName">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element maxOccurs="unbounded" name="Name"
type="xs:string" />
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                          <xs:element minOccurs="0" name="LocalizedComment">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element maxOccurs="unbounded" name="Comment"
type="xs:string" />
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                          <xs:element minOccurs="0" name="ContentSettings">
                            <xs:complexType>
                              <xs:attribute name="Guid" type="xs:string" use="required" />
                              <xs:attribute name="ExpiryAgeLimit" type="xs:unsignedShort"
use="required" />
                              <xs:attribute name="MessageClass" type="xs:string"
use="required" />
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                        <xs:attribute name="Guid" type="xs:string" use="required" />
                        <xs:attribute name="Name" type="xs:string" use="required" />
                        <xs:attribute name="Comment" type="xs:string" />
                        <xs:attribute name="Type" type="xs:string" use="required" />
                        <xs:attribute name="MustDisplayComment" type="xs:string"
use="required" />
                        <xs:attribute name="IsVisible" type="xs:string" use="required" />
                        <xs:attribute name="OptedInto" type="xs:string" use="required" />
                      </xs:complexType>
                    </xs:element>
                    <xs:element maxOccurs="unbounded" name="ArchiveTag">
                      <xs:complexType>
                        <xs:sequence minOccurs="0">
                          <xs:element minOccurs="0" name="LocalizedName">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element maxOccurs="unbounded" name="Name"
type="xs:string" />
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                          <xs:element minOccurs="0" name="LocalizedComment">
                            <xs:complexType>
```

```
                            <xs:sequence>
                              <xs:element maxOccurs="unbounded" name="Comment"
type="xs:string" />
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                        <xs:element minOccurs="0" name="ContentSettings">
                          <xs:complexType>
                            <xs:attribute name="Guid" type="xs:string" use="required" />
                            <xs:attribute name="ExpiryAgeLimit" type="xs:unsignedShort"
use="required" />
                            <xs:attribute name="MessageClass" type="xs:string"
use="required" />
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                      <xs:attribute name="Guid" type="xs:string" use="required" />
                      <xs:attribute name="Name" type="xs:string" use="required" />
                      <xs:attribute name="Comment" type="xs:string" />
                      <xs:attribute name="Type" type="xs:string" use="required" />
                      <xs:attribute name="MustDisplayComment" type="xs:string"
use="required" />
                      <xs:attribute name="IsVisible" type="xs:string" use="required" />
                      <xs:attribute name="OptedInto" type="xs:string" use="required" />
                    </xs:complexType>
                  </xs:element>
                  <xs:element maxOccurs="1" name="DefaultArchiveTag">
                    <xs:complexType>
                      <xs:sequence minOccurs="0">
                        <xs:element minOccurs="0" name="LocalizedName">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element maxOccurs="unbounded" name="Name"
type="xs:string" />
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                        <xs:element minOccurs="0" name="LocalizedComment">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element maxOccurs="unbounded" name="Comment"
type="xs:string" />
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                        <xs:element minOccurs="0" name="ContentSettings">
                          <xs:complexType>
                            <xs:attribute name="Guid" type="xs:string" use="required" />
                            <xs:attribute name="ExpiryAgeLimit" type="xs:unsignedShort"
use="required" />
                            <xs:attribute name="MessageClass" type="xs:string"
use="required" />
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                      <xs:attribute name="Guid" type="xs:string" use="required" />
                      <xs:attribute name="Name" type="xs:string" use="required" />
                      <xs:attribute name="Comment" type="xs:string" />
                      <xs:attribute name="Type" type="xs:string" use="required" />
```

```
                        <xs:attribute name="MustDisplayComment" type="xs:string"
use="required" />
                        <xs:attribute name="IsVisible" type="xs:string" use="required" />
                        <xs:attribute name="OptedInto" type="xs:string" use="required" />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="version" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Info: The parent element of the **Data** element.

Data: A child of the **Info** element. A container element for the other retention policy settings.

RetentionHold: A child of the **Data** element. Includes a series of sub-elements that describe the retention policy in effect for this user. Has the following attributes:

Attribute name	Description
Enabled	Indicates whether retention is enabled for a user. If it is enabled, then the client can display specific information to indicate that messages will not expire. This value is configured by a server administrator. The value MUST be "True" or "False".
RetentionComment	Contains a string used to communicate information regarding the retention policy feature to the user. This value is configured by a server administrator. The client application SHOULD display this to the user. The string length MUST be between 0 and 1,024 characters.
RetentionUrl	Contains a link to a Web page that contains more information about the retention policy. This value is configured by a server administrator. The client application SHOULD display this to the user. The string length MUST be between 0 and 2,048 characters.

PolicyTag: A child of the **Data** element. Stores information about the retention policy. The information is stored in the attributes defined in the following table.

Attribute name	Description
Guid	A GUID value that provides a unique identity for the policy.
Name	The name of the policy tag. This value is configured by a server administrator. The value MUST be unique; no two policy tags or archive tags can have the same name. The string length MUST be between 1 and 264 characters.
Comment	A brief description of the policy tag. This value is configured by a server administrator. The string length MUST be between 1 and 264 characters.
Type	The type of policy tag. This value is configured by a server administrator. The

Attribute name	Description
	possible values are as follows: ▪ All ▪ Calendar ▪ Contacts ▪ ConversationHistory ▪ DeletedItems ▪ Drafts ▪ Inbox ▪ JunkEmail ▪ Journal ▪ Notes ▪ Outbox ▪ Personal ▪ RssSubscriptions ▪ SentItems ▪ SyncIssues ▪ Tasks Policy tags with a type of "All" apply to all items that do not have an applicable policy tag. Policy tags of type "Personal" are applied explicitly by the user. All other tags are applied to the corresponding special folder.
MustDisplayComment	The client application MAY use this setting to determine whether the user can hide the retention comment. The value SHOULD be "True" or "False". If the value is "True", the client MAY NOT allow the user to hide the retention comment.
IsVisible	The client application MUST use this setting to determine whether the policy tag is displayed in the list of tags available to the user. The value SHOULD be "True" or "False". If the value is "True", the client MUST display the policy tag in the list of tags available to the user.
OptedInto	Indicates whether the user voluntarily chose this tag for the **mailbox** or whether it was applied to the mailbox by the server administrator. The value SHOULD be "True" or "False". A value of "True" SHOULD<15> indicate that the user voluntarily chose this tag.

LocalizedName: A child of the **PolicyTag** element. It is also a child of the **ArchiveTag** and **DefaultArchiveTag** elements. Can contain multiple **Name** elements as children. Each contains the ISO language code followed by a colon (":") and the localized name of the tag. The string length MUST be between 0 and 264 characters.

LocalizedComment: A child of the **PolicyTag** element. It is also a child of the **ArchiveTag** and **DefaultArchiveTag** elements. Can contain multiple **Comment** elements as children. Each contains the ISO language code followed by a colon (":") and the localized comment. The string length MUST be between 0 and 264 characters.

ContentSettings: A child of the **PolicyTag** element. It is also a child of the **ArchiveTag** and **DefaultArchiveTag** elements. Stores the settings that control when items expire and the types of items that expire. The settings are stored in the following attributes.

Attribute name	Description
Guid	A GUID value that provides a unique identity for the content setting.
ExpiryAgeLimit	The number of days after which an item SHOULD expire under this policy. This value is configured by a server administrator. The value MUST be between 0 and 24,855.
MessageClass	The **message classes** that this content setting applies to. This value is configured by a server administrator. This value can be any string that represents a message class. The value "*" represents any message class. The string length MUST be between 1 and 1,023 characters.

ArchiveTag: A child of the **Data** element. Stores information about an archive policy. The information is stored in attributes and child elements identical to those on the **PolicyTag** element. The value of the **ArchiveTag** element SHOULD NOT be displayed by the client application in the archive mailbox. It SHOULD be displayed only in the primary mailbox.

DefaultArchiveTag: A child of the **Data** element. Stores information about the default archive policy. The information is stored in attributes and child elements identical to those on the **PolicyTag** element. The value of the **DefaultArchiveTag** element SHOULD NOT be displayed by the client application in the archive mailbox. It SHOULD be displayed only in the primary mailbox.

2.2.6 View Definitions

The client and server MAY<16> store view settings as view definitions. The format of the view definitions, as well as which settings are included, is specified in sections 2.2.6.1 through 2.2.6.2.

View definitions can be created by the client to make view settings available to the server. These settings consist of column descriptions, including column header names and sizes, groupings, **sort orders (3)**, and an optional restriction (1). The data is stored in several stream (2) properties in an FAI message.

A message that contains view definitions MUST be an FAI message in the folder where the view is used. The message MUST have the following properties set on it, and the value of each property MUST meet the following criteria:

The message has the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) set and the value of the property is "IPM.Microsoft.FolderDesign.NamedView".

The message has the **PidTagViewDescriptorVersion** property ([MS-OXPROPS] section 2.1046) set and the value of the property is 0x00000008.

The message has the **PidTagViewDescriptorName** property ([MS-OXPROPS] section 2.1044) set and the value of the property is a non-empty string.

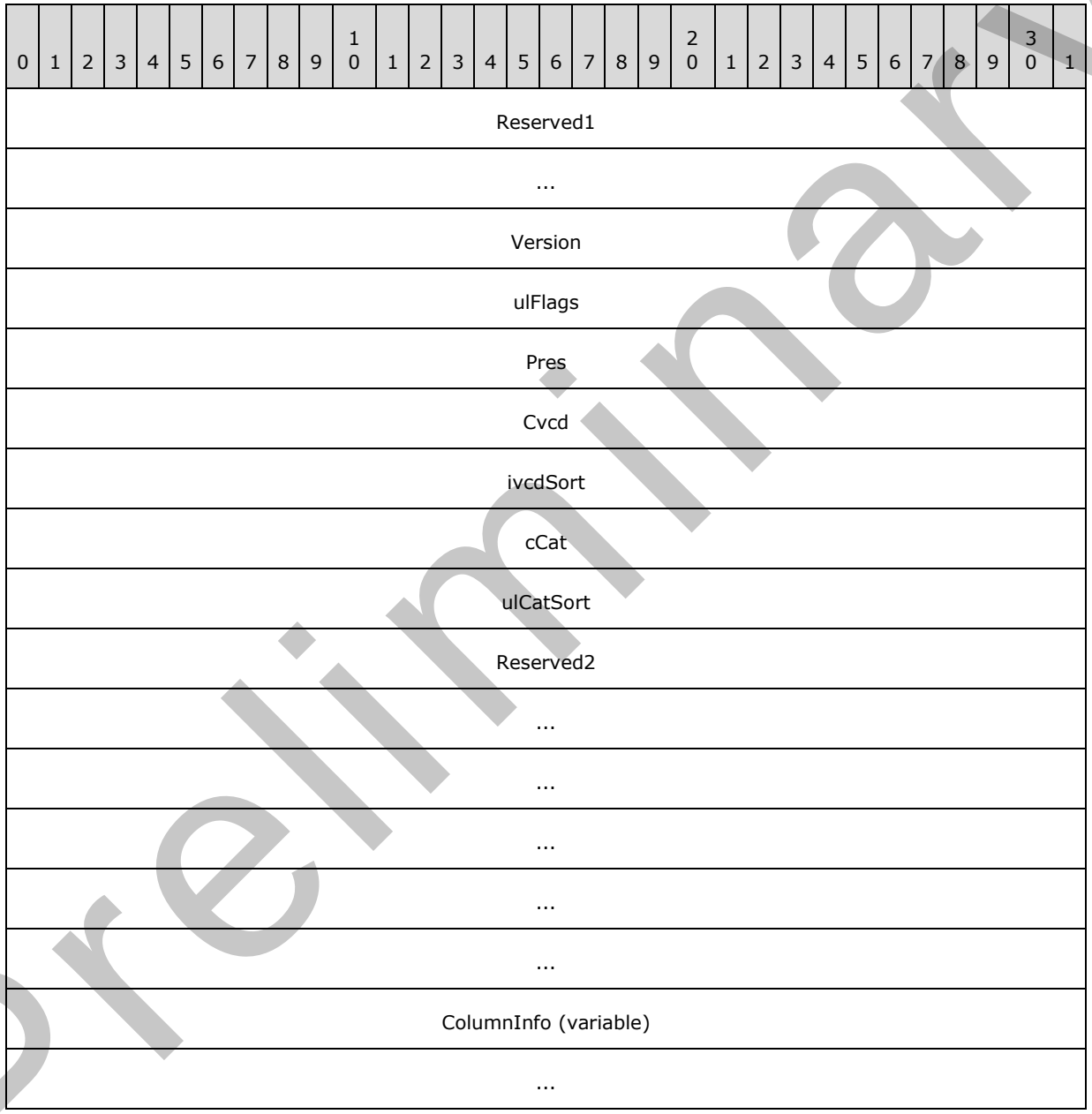
The view definitions MUST be stored as a binary stream (2) in the **PidTagViewDescriptorBinary** property (section 2.2.6.1) of the message. The column headers are stored in the **PidTagViewDescriptorStrings** property (section 2.2.6.2) on the message as strings using the

current **code page** of the client. Section 2.2.6.1 and section 2.2.6.2 specify the packet format of these two properties respectively.

2.2.6.1 PidTagViewDescriptorBinary Property

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

View definitions, as specified in section 2.2.6, MUST be stored in the **PidTagViewDescriptorBinary** property ([MS-OXPROPS] section 2.1043) of the message.<17> The property is in binary format, and the packet structure is specified as follows.



RestrictionInfo (variable)
...

Reserved1 (8 bytes): This field MUST exist. The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

Version (4 bytes): This field MUST exist. This is a fixed value of 0x00000008.

ulFlags (4 bytes): This field MUST exist. This specifies the sort order (3) of the sorted column. The value of this field MUST be one of the following.

Value	Meaning
0x00000000	Ascending sort order (3)
0x00000002	Descending sort order (3)

The index of the sorted column is indicated in the **ivcdSort** field in the packet.

Pres (4 bytes): This field MUST exist. This field is filled with arbitrary value by the client and SHOULD NOT be used by the server.

Cvcd (4 bytes): This field MUST exist. This field specifies the number of **ColumnInfo** fields that are stored in this packet.

ivcdSort (4 bytes): This field MUST exist. The value of this field MUST be one of the following:

0 through (**Cvcd**-1): This value is an index into the **ColumnInfo** fields. The table MUST be sorted by that column. The sort order (3), ascending or descending, MUST be specified in the **ulFlags** field.

0xFFFFFFFF: The table MUST be arranged by **conversation**.

cCat (4 bytes): This field MUST exist. This field specifies the number of "group by" columns that are stored in **ColumnInfo** fields. The minimum value for this field is 0. The maximum value is either 4 or the value of the **Cvcd** field, whichever is least.

ulCatSort (4 bytes): This field MUST exist. This field uses bit flags to specify the ascending or descending order of the "group by" columns. The flags are specified as follows. In each case, if the flag is not set, the "group by" column is in descending order.

Flag	Meaning
0x00000001	If this flag is set, the first "group by" column is in ascending order.
0x00000002	If this flag is set, the second "group by" column is in ascending order.
0x00000004	If this flag is set, the third "group by" column is in ascending order.
0x00000008	If this flag is set, the fourth "group by" column is in ascending order.

Reserved2 (24 bytes): This field MUST exist. The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

ColumnInfo (variable): Data type: array of **ColumnPacket** structures, as specified in section 2.2.6.1.1. The number of **ColumnPacket** structures contained in this field is indicated by the **Cvcd** field.

This field MUST exist. This is where all the column information is stored, including "blank" column, "group by" columns, "visible" columns, and "order by" column. The count of the columns is specified by the **Cvcd** field in the packet.

The columns are stored in the following sequence in the packet:

The "blank" column: This is a single column that MUST have the following settings.

Field name	Value
PropertyType	0x0001
PropertyID	0x0004
Cx	0x00000007
Flags	0x00000028 (VCDF_BITMAP | VCDF_NOT_SORTABLE)
Kind	0x00000000
ID	0x00000004

The "group by" columns: The number of the "group by" columns MUST be stored in **cCat** field in the packet. Each bit in the **ulCatSort** field MUST specify whether the corresponding "group by" column is in ascending or descending order.

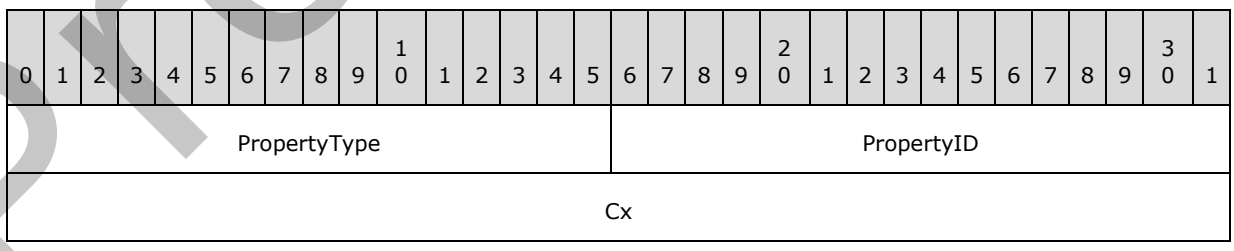
The "visible" columns: All columns that MUST be visible to users excluding the "group by" columns.

The "order by" column: If the sorted column is not a "group by" or "visible" column, it MUST be stored here.

RestrictionInfo (variable): Data type: **RestrictionPacket** structure, as specified in section 2.2.6.1.2. This field is where the restriction (1) of the table view MUST be stored.

2.2.6.1.1 ColumnPacket Structure

The **ColumnPacket** structure MUST contain the information of a single column, including the property ID, **property type**, and display attributes. The **ColumnPacket** structure contains the following fields.



Reserved1
Flags
Reserved2
...
...
Kind
ID
Guid
...
...
...
BufferLength
Buffer (variable)
...

PropertyType (2 bytes): This field MUST exist. This field specifies the type of the property, as specified in section 2.2.6.1.3.

PropertyID (2 bytes): This field MUST exist. This field has the same value as the **ID** field. If the value of the **ID** field does not fit into a **WORD** ([MS-DTYP]), the value MUST be truncated and the two least significant bytes MUST be stored in this field.

Cx (4 bytes): This field MUST exist. This field specifies the column width in pixels.

Reserved1 (4 bytes): This field MUST exist. The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

Flags (4 bytes): This field MUST exist. This field contains column descriptor flags. The bit setting and its meaning are listed in the following table.

Flag name	Value	Meaning
VCDF_RIGHT_JUSTIFY	0x00000001	Column is right justified. This flag is mutually exclusive with the **VCDF_CENTER_JUSTIFY** flag.
VCDF_CENTER_JUSTIFY	0x00000002	Column is center justified. This flag is mutually

Flag name	Value	Meaning
		exclusive with the **VCDF_RIGHT_JUSTIFY** flag.
VCDF_BITMAP	0x00000008	Column header is in bitmap format.
VCDF_NOT_SORTABLE	0x00000020	Column is not sortable. This flag is mutually exclusive with the **VCDF_SORTDESCENDING** flag and the **VCDF_SORTDLG** flag.
VCDF_SORTDESCENDING	0x00000040	Column is sorted in descending order. This flag is mutually exclusive with the **VCDF_NOT_SORTABLE** flag.
VCDF_MOVEABLE	0x00000100	Clients and servers MUST ignore this setting.
VCDF_COLUMNSDLG	0x00000200	Clients and servers MUST ignore this setting.
VCDF_SORTDLG	0x00000400	Column MUST be able to be sorted. This flag is mutually exclusive with the **VCDF_NOT_SORTABLE** flag.
VCDF_GROUPDLG	0x00000800	Column MUST be able to be grouped.
VCDF_NAMEDPROP	0x00001000	The optional **Guid** field MUST be included in the packet. If **Kind** is KindString, then the **BufferLength** and **Buffer** fields MUST also be included in the packet.
VCDF_RCOLUMNSDLG	0x00002000	Clients and servers MUST ignore this setting.
VCDF_MULTIVALUED	0x00004000	Specifies whether the column **PropertyType** field MUST include the **VCDF_MULTIVALUED** flag ([MS-OXCDATA] section 2.11.1.3).

Reserved2 (12 bytes): This field MUST exist. The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

Kind (4 bytes): This field MUST exist. The field contains one of the following values.

Value name	Value	Meaning
KindID	0x00000000	The property uses an integer identifier.
KindString	0x00000001	The property uses a string identifier.

ID (4 bytes): This field MUST exist. If the **VCDF_NAMEDPROP** flag is not set in the **Flags** field, this field contains the property ID of the column. If the **VCDF_NAMEDPROP** flag is set in the **Flags** field, and the value of the **Kind** field is "KindID", this field contains the integer ID that MUST be used with the **RopGetPropertyIdsFromNames** **remote operation (ROP)** ([MS-OXCROPS] section 2.2.8.1) to translate the **named property** into a property ID. If the **VCDF_NAMEDPROP** flag is set and the value of **Kind** is "KindString", the application can fill this field with any value when writing the stream (2) and MUST ignore the value of this field when reading the stream (2).

Guid (16 bytes): If the **VCDF_NAMEDPROP** flag is set in the **Flags** field, this field contains the GUID that MUST be used with the **RopGetPropertyIdsFromNames** ROP to translate the

named property into a property ID. If the **VCDF_NAMEDPROP** flag is not set, the application MUST omit this field.

BufferLength (4 bytes): If the **VCDF_NAMEDPROP** flag is set in the **Flags** field and the value of the **Kind** field is "KindString", this field contains the length of the **Buffer** field in bytes, including the Unicode terminating null character (0x0000). Otherwise, the application MUST omit this field.

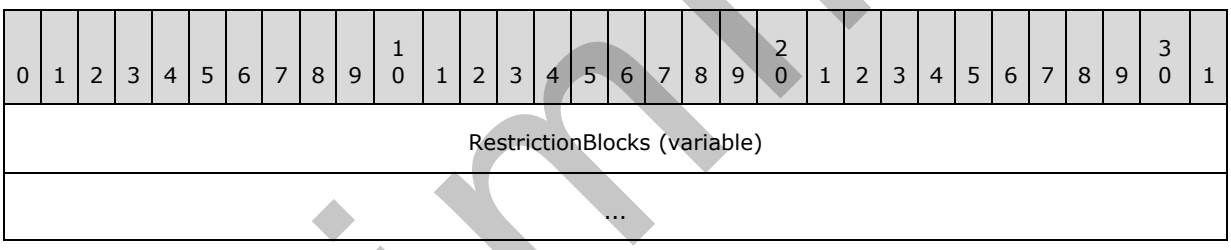
Buffer (variable): If the **VCDF_NAMEDPROP** flag is set in the **Flags** field and the value of the **Kind** field is "KindString", this field contains the Unicode string that MUST be used with the **RopGetPropertyIdsFromNames** ROP to translate the named property into a property ID. Otherwise, the application MUST omit this field. This field includes a Unicode terminating null character (0x0000).

2.2.6.1.2 RestrictionPacket Structure

The **RestrictionPacket** structure contains the restrictions (1) that are used to evaluate the **contents table** of the folder. Only those rows that evaluate to a value of **TRUE** MUST be displayed.

The **RestrictionPacket** structure is recursively built up by the restriction structures that are specified in section 2.2.6.1.2.1 through section 2.2.6.1.2.11. To determine the size of the **RestrictionPacket** structure, the application parses each restriction structure recursively if necessary. The restrictions (1) are stored in a special format, which is different from the format specified for restrictions (1) in [MS-OXCDATA] section 2.12.

The **RestrictionPacket** structure starts with a single restriction structure. The type of the restriction determines whether other restrictions will follow each restriction, as specified in the following description.



RestrictionBlocks (variable): This field contains one or more of the restriction structures that are specified in section 2.2.6.1.2.1 through section 2.2.6.1.2.11. From the restriction type, specified in the **RestrictionType** field of each restriction structure, the application can determine whether the restriction structure contains subrestrictions, which are are stored sequentially in the **RestrictionPacket** structure. The server parses each restriction structure recursively, if necessary, to complete reading one sequence of restriction structures.

The following table specifies the values for the **RestrictionType** field and the associated restriction.

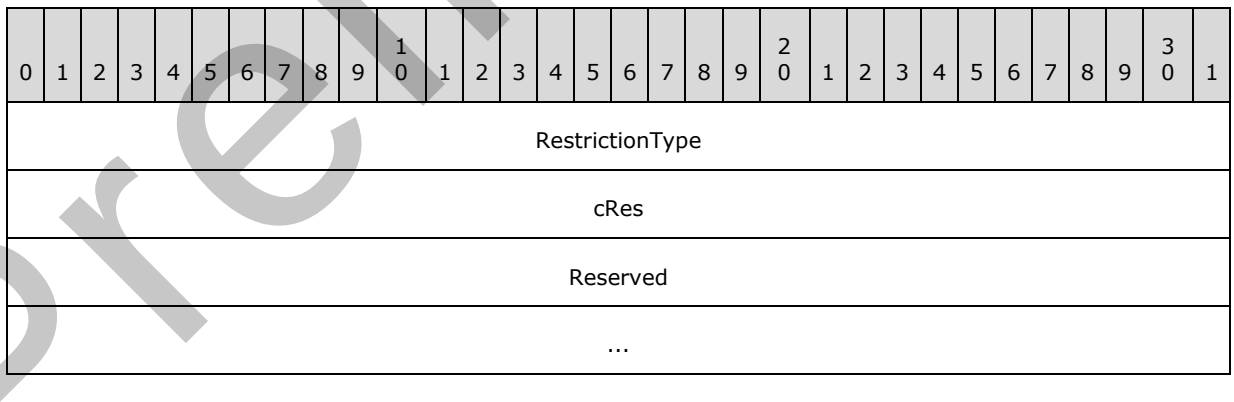
Value of RestrictionType field	Numeric value	Meaning
RES_AND	0x00000000	Indicates a **LogicalAndRestriction** structure (section 2.2.6.1.2.1). This structure has subrestrictions that follow it.
RES_OR	0x00000001	Indicates a **LogicalOrRestriction** structure (section

Value of RestrictionType field	Numeric value	Meaning
		2.2.6.1.2.2). This structure has subrestrictions that follow it.
RES_NOT	0x00000002	Indicates a **LogicalNotRestriction** structure (section 2.2.6.1.2.3). This structure has a single subrestriction that follows it.
RES_CONTENT	0x00000003	Indicates a **ContentRestriction** structure (section 2.2.6.1.2.4).
RES_PROPERTY	0x00000004	Indicates a **PropertyRestriction** structure (section 2.2.6.1.2.5).
RES_COMPAREPROPS	0x00000005	Indicates a **ComparePropsRestriction** structure (section 2.2.6.1.2.6).
RES_BITMASK	0x00000006	Indicates a **BitmaskRestriction** structure (section 2.2.6.1.2.7).
RES_SIZE	0x00000007	Indicates a **SizeRestriction** structure (section 2.2.6.1.2.8).
RES_EXIST	0x00000008	Indicates a **ExistRestriction** structure (section 2.2.6.1.2.9).
RES_SUBRESTRICTION	0x00000009	Indicates a **SubObjectRestriction** structure (section 2.2.6.1.2.10). This structure has a single subrestriction that follows it.
RES_COMMENT	0x0000000A	Indicates a **CommentRestriction** structure (section 2.2.6.1.2.11). This structure has a single subrestriction that follows it.

2.2.6.1.2.1 LogicalAndRestriction Structure

The **LogicalAndRestriction** structure is used to join a group of two or more subrestrictions by using a logical **AND** operation. The result of the **AND** operation is TRUE if all of the subrestrictions evaluate to TRUE. The result is FALSE if any subrestriction evaluates to FALSE.

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



SubCondition (variable)
...

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_AND** (0x00000000) for the **LogicalAndRestriction** structure.

cRes (4 bytes): Specifies the number of subrestrictions contained in the **SubCondition** field.

Reserved (8 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

SubCondition (variable): This field contains the subrestrictions that make up the **LogicalAndRestriction** structure. Each subrestriction MUST be one of the structures that is specified in section 2.2.6.1.2.1 through section 2.2.6.1.2.11.

2.2.6.1.2.2 LogicalOrRestriction Structure

The **LogicalOrRestriction** structure is used to join a group of two or more subrestrictions by using a logical **OR** operation. The result of the **OR** operation is TRUE if any of the subrestrictions evaluates to TRUE. The result is FALSE if all subrestrictions evaluate to FALSE.

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

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
RestrictionType																															
cRes																															
Reserved																															
...																															
SubCondition (variable)																															
...																															

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_OR** (0x00000001) for the **LogicalOrRestriction** structure.

cRes (4 bytes): This field specifies the number of subrestrictions contained in the **SubCondition** field.

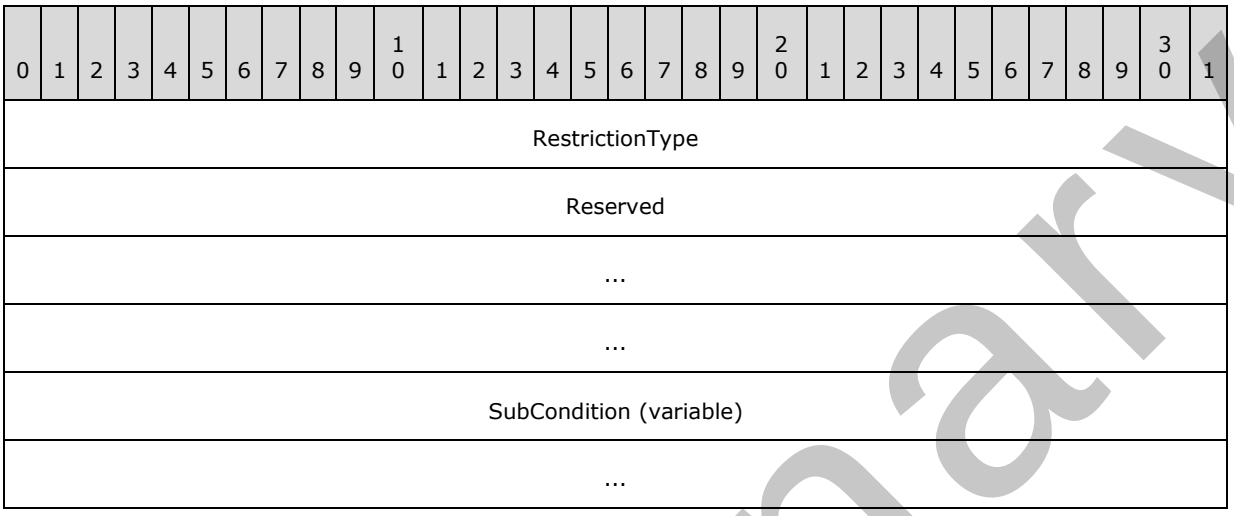
Reserved (8 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

SubCondition (variable): This field contains the subrestrictions that make up the **LogicalOrRestriction** structure. Each subrestriction MUST be one of the structures that is specified in section 2.2.6.1.2.1 through section 2.2.6.1.2.11.

2.2.6.1.2.3 LogicalNotRestriction Structure

The **LogicalNotRestriction** structure is used to apply a logical **NOT** operation to one subrestriction. The result is TRUE if the child condition evaluates to FALSE and FALSE otherwise.

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



RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_NOT** (0x00000002) for the **LogicalNotRestriction** structure.

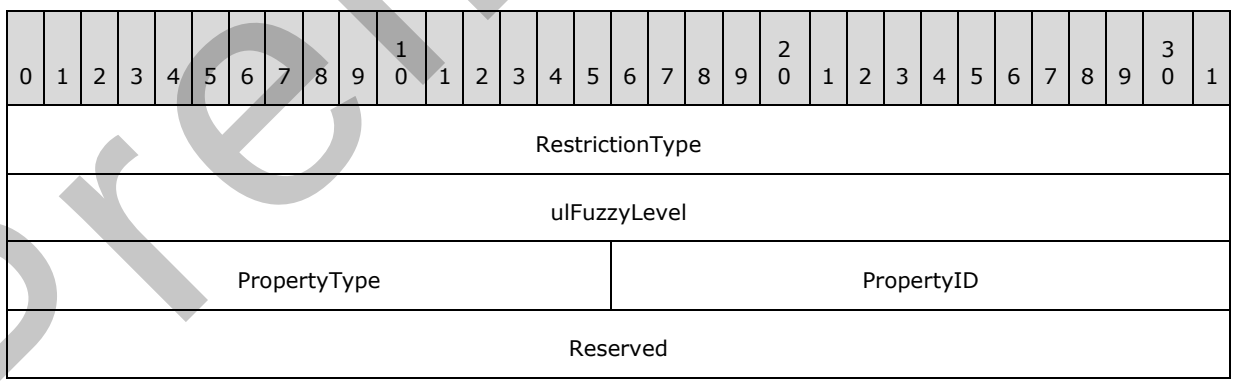
Reserved (12 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

SubCondition (Variable): This field contains a single subrestriction that makes up this structure. The subrestriction MUST be one of the structures that is specified in section 2.2.6.1.2.1 through section 2.2.6.1.2.11.

2.2.6.1.2.4 ContentRestriction Structure

The **ContentRestriction** structure is used to search for properties with values that match the value of a specified property.

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



PropValueNum
PropValue (variable)

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_CONTENT** (0x00000003) for the **ContentRestriction** structure.

ulFuzzyLevel (4 bytes): This field specifies flags that control the behavior of the string comparisons that are used to evaluate the restriction (1). The lower 16 bits of the fuzzy level are mutually exclusive. All possible values for this field are listed in the following table.

Flag name	Value	Meaning
FL_FULLSTRING	0x00000000	To match, the search string MUST be the same as the value of the property.
FL_SUBSTRING	0x00000001	To match, the search string MUST be contained anywhere within the property.
FL_PREFIX	0x00000002	To match, the search string MUST appear at the beginning of the property. The two strings MUST be compared only up to the length of the search.

The upper 16 bits of the fuzzy level can be set to the following values and can be combined using the logical **OR** operation.

Flag name	Value	Meaning
FL_IGNORECASE	0x00010000	The comparison MUST be made without considering the case.
FL_IGNORENONSPACE	0x00020000	The comparison MUST ignore Unicode-defined nonspacing characters.
FL_LOOSE	0x00040000	The comparison can result in a match whenever possible, ignoring case and nonspacing characters. The interpretation of this flag is left at the discretion of the algorithm that implements the restriction (1).

PropertyType (2 bytes): This field specifies the type of the property. Valid types are specified in section 2.2.6.1.3.

PropertyID (2 bytes): This field specifies the **property ID** of the property.

Reserved (4 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

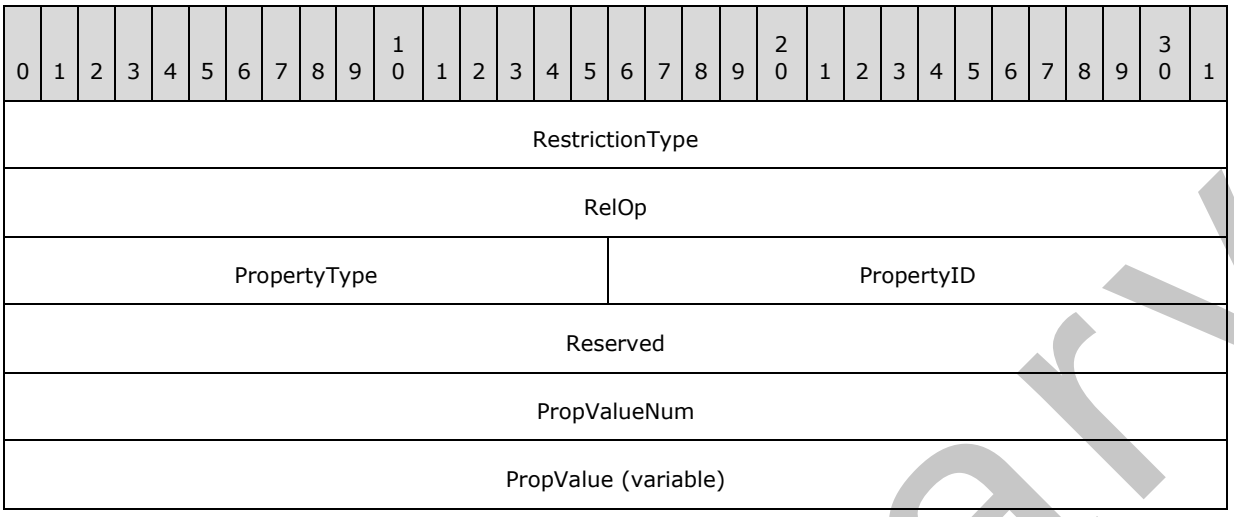
PropValueNum (4 bytes): This field MUST be set to 0x00000001.

PropValue (variable): A **PropertyValue** structure, as specified in section 2.2.6.1.4, that specifies the value to be matched.

2.2.6.1.2.5 PropertyRestriction Structure

The **PropertyRestriction** structure is used to compare the value of a property with a constant.

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



RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_PROPERTY** (0x00000004) for the **PropertyRestriction** structure.

RelOp (4 bytes): This field specifies the relational operator to be used in the comparison. The value MUST be one of the following.

Relational operator name	Value	Meaning
RELOP_LT	0x00000000	The condition evaluates to TRUE if the value of the property is less than the constant value.
RELOP_LE	0x00000001	The condition evaluates to TRUE if the value of the property is less than or equal to the constant value.
RELOP_GT	0x00000002	The condition evaluates to TRUE if the value of the property is greater than the constant value.
RELOP_GE	0x00000003	The condition evaluates to TRUE if the value of the property is greater than or equal to the constant value.
RELOP_EQ	0x00000004	The condition evaluates to TRUE if the value of the property is equal to the constant value.
RELOP_NE	0x00000005	The condition evaluates to TRUE if the value of the property is not equal to the constant value.

PropertyType (2 bytes): This field specifies the type of the property. Valid types are specified in section 2.2.6.1.3.

PropertyID (2 bytes): This field specifies the property ID of the property.

Reserved (4 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

PropValueNum (4 bytes): This field MUST be set to 0x00000001.

PropValue (variable): A **PropertyValue** structure, as specified in section 2.2.6.1.4, that specifies the value to be compared.

2.2.6.1.2.6 ComparePropsRestriction Structure

The **ComparePropsRestriction** structure is used to compare the values of two properties by using a relational operator.

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

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_COMPAREPROPS** (0x00000005) for the **ComparePropsRestriction** structure.

RelOp (4 bytes): This field specifies the relational operator that is to be used in the comparison. Valid values for this field are the same as those specified for the **RelOp** field of the **PropertyRestriction** structure in section 2.2.6.1.2.5.

PropertyType1 (2 bytes): This field specifies the type of the first property. Valid types are specified in section 2.2.6.1.3.

PropertyID1 (2 bytes): This field specifies the property ID of the first property.

PropertyType2 (2 bytes): This field specifies the type of the second property. The type of the second property MUST match the type of the first property.

PropertyID2 (2 bytes): This field specifies the property ID of the second property.

2.2.6.1.2.7 BitmaskRestriction Structure

The **BitmaskRestriction** structure is used to perform a bitwise **AND** operation on the value of the property.

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

PropertyType	PropertyID
Mask	

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_BITMASK** (0x00000006) for the **BitmaskRestriction** structure.

RelOp (4 bytes): This field specifies the relational operator that is to be used in the search. The value MUST be one of the following.

Relational operator name	Value	Meaning
BMR_EQZ	0x00000000	Perform a bitwise **AND** operation between the value of the **Mask** field and the value of the property identified by the **PropertyID** and **PropertyType** fields. The comparison returns TRUE if the result of the operation is zero.
BMR_NEZ	0x00000001	Perform a bitwise **AND** operation between the value of the **Mask** field and the value of the property identified by the **PropertyID** and **PropertyType** fields. The comparison returns TRUE if the result of the operation is not zero.

PropertyType (2 bytes): This field specifies the type of the property. Valid types are specified in section 2.2.6.1.3.

PropertyID (2 bytes): This field specifies the property ID of the property.

Mask (4 bytes): This field specifies the bitmask that the application MUST use in a bitwise **AND** operation with the value of the property when performing the search.

2.2.6.1.2.8 SizeRestriction Structure

The **SizeRestriction** structure is used to test the size of a property value.

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

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
RestrictionType																															
RelOp																															
PropertyType																				PropertyID											
Size																															

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_SIZE** (0x00000007) for the **SizeRestriction** structure.

RelOp (4 bytes): This field specifies the relational operator that is to be used in the search. Valid values for this field are the same as those specified for the **RelOp** field of the **PropertyRestriction** structure in section 2.2.6.1.2.5.

PropertyType (2 bytes): This field specifies the type of the property. Valid types are specified in section 2.2.6.1.3.

PropertyID (2 bytes): This field specifies the property ID of the property.

Size (4 bytes): This field specifies the size, in bytes, that MUST be compared with the size of the value of this property when performing the search.

2.2.6.1.2.9 ExistRestriction Structure

The **ExistRestriction** structure is used to test whether a particular property exists on a message.

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

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_EXIST** (0x00000008) for the **ExistRestriction** structure.

Reserved1 (4 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

PropertyType (2 bytes): This field specifies the type of the property. Valid types are specified in section 2.2.6.1.3.

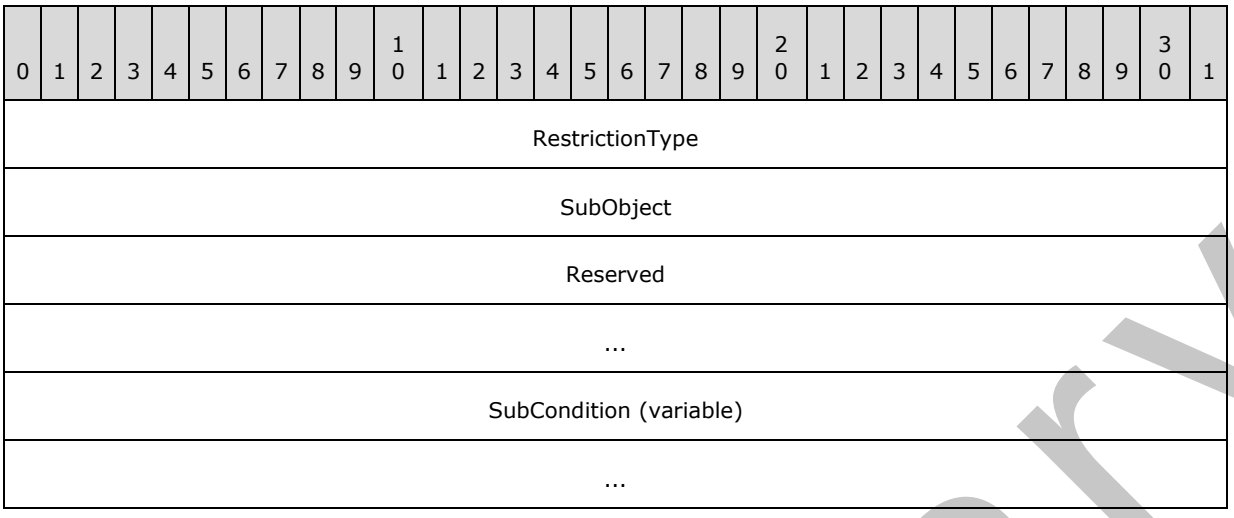
PropertyID (2 bytes): This field specifies the property ID of the property.

Reserved2 (4 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

2.2.6.1.2.10 SubObjectRestriction Structure

The **SubObjectRestriction** structure is used to test properties on the attachment or **recipient table** of a message.

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



RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_SUBRESTRICTION** (0x00000009) for the **SubObjectRestriction** structure.

SubObject (4 bytes): The application MUST use one of the following values for this field.

Value	Meaning
0x0E12000D	Apply the condition to the recipient table of a message.
0x0E13000D	Apply the condition to the **attachments table** of a message.

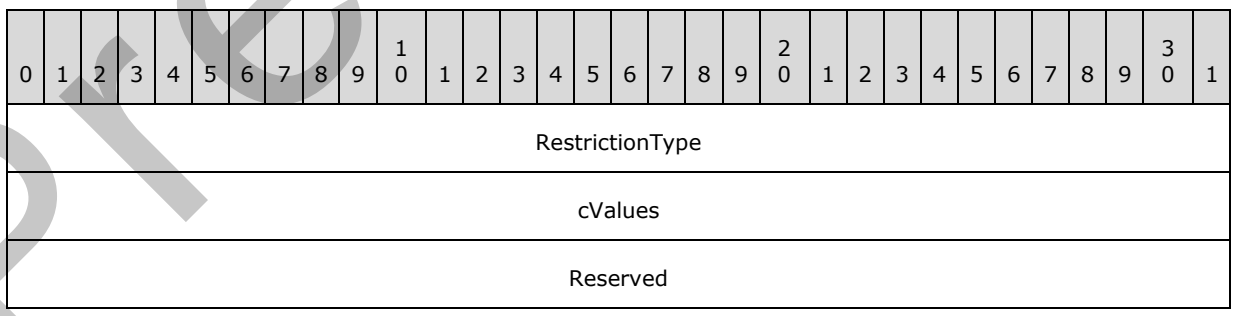
Reserved (8 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

SubCondition (variable): This field contains a single subrestriction that makes up this structure. The subrestriction MUST be one of the structures that is specified in section 2.2.6.1.2.1 through section 2.2.6.1.2.11.

2.2.6.1.2.11 CommentRestriction Structure

The **CommentRestriction** structure, unlike other restriction structures, is not evaluated and is used only for reference by the application. The comment condition is used to keep additional application-specific information with the restriction (1) in the form of an arbitrary list of **property tag** and value pairs.

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



...
PropValueNum
PropTags (variable)
...
PropValues (variable)
...
SubCondition (variable)
...

RestrictionType (4 bytes): This field specifies the type of restriction and MUST be set to **RES_COMMENT** (0x0000000A) for the **CommentRestriction** structure.

cValues (4 bytes): This field MUST have the same value as the **PropValueNum** field.

Reserved (8 bytes): The application can fill this field with any value when writing the stream (2). The application MUST ignore the value of this field when reading the stream (2).

PropValueNum (4 bytes): This field is specifies the number of structures contained in the **PropTags** field and the **PropValues** field.

PropTags (variable): An array of **PropertyTag** structures, each of which specifies the property tag of a property that is specified in the **PropValues** field. For details about the **PropertyTag** structure, see [MS-OXCDATA] section 2.9. The valid property types for restrictions are specified in section 2.2.6.1.3. The number of **PropertyTag** structures in the array is specified by the **PropValueNum** field.

PropValues (variable): An array of **PropertyValue** structures. For details about the **PropertyValue** structure, see section 2.2.6.1.4. The number of structures is specified by the **PropValueNum** field.

SubCondition (variable): This field contains a single subrestriction that makes up the **CommentRestriction** structure. The subrestriction MUST be one of the structures that is specified in section 2.2.6.1.2.1 through section 2.2.6.1.2.11.

2.2.6.1.3 Valid Property Types

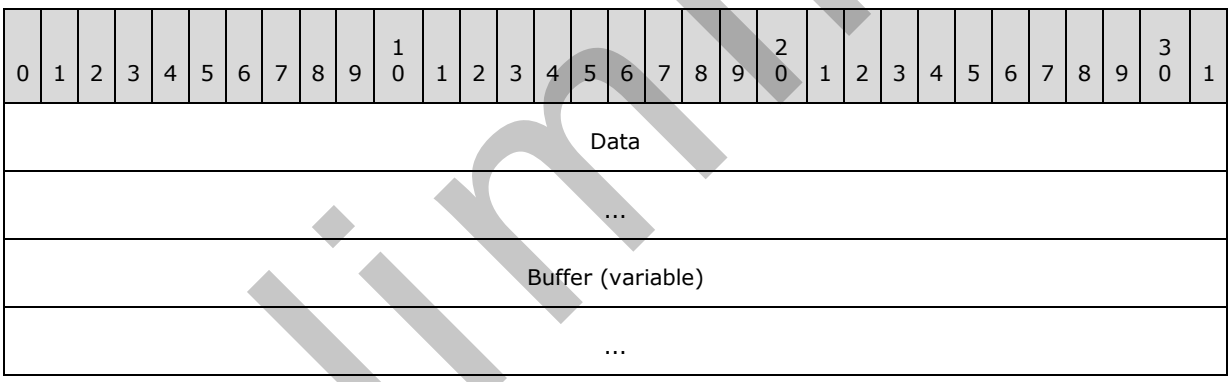
The following property types are valid for a **ColumnPacket** structure, as specified in section 2.2.6.1.1, or for a **RestrictionPacket** structure, as specified in section 2.2.6.1.2, that specifies a property.

Property data type	Type ID
PtypInteger16	0x0002
PtypInteger32	0x0003

Property data type	Type ID
PtypFloating32	0x0004
PtypFloating64	0x0005
PtypCurrency	0x0006
PtypFloatingTime	0x0007
PtypErrorCode	0x000A
PtypBoolean	0x000B
PtypInteger64	0x0014
PtypString	0x001F
PtypTime	0x0040
PtypGuid	0x0048
PtypBinary	0x0102

2.2.6.1.4 PropertyValue Structure

The **PropertyValue** structure specifies the value of a property in a **ContentRestriction** structure, as specified in section 2.2.6.1.2.4, a **PropertyRestriction** structure, as specified in section 2.2.6.1.2.5, or a **CommentRestriction** structure, as specified in section 2.2.6.1.2.11.



Data (8 bytes): The format of this field depends on the property type that is specified in the **ContentRestriction**, **PropertyRestriction**, or **CommentRestriction** structures. The value of this field is read from the beginning of the field. In cases where the size of the data is less than the size of the **Data** field, the remaining bytes MUST be ignored. The following table lists the size and format of the value by property type.

Property type	Value size	Value format
PtypInteger16	2 bytes	Signed integer
PtypInteger32	4	Signed integer

Property type	Value size	Value format
	bytes	
PtypFloating32	4 bytes	Floating point number
PtypFloating64	8 bytes	Floating point number
PtypCurrency	8 bytes	Signed integer
PtypFloatingTime	8 bytes	Floating point number
PtypErrorCode	4 bytes	SCODE error code
PtypBoolean	2 bytes	WORD
PtypInteger64	8 bytes	Signed integer
PtypString	0 bytes	The string value MUST be stored separately in the **PidTagViewDescriptorStrings** property, as specified in section 2.2.6.2. The value of the **Data** field MUST be ignored.
PtypTime	8 bytes	Unsigned integer
PtypGuid	0 bytes	The value of the **Data** field MUST be ignored.
PtypBinary	4 bytes	Unsigned integer

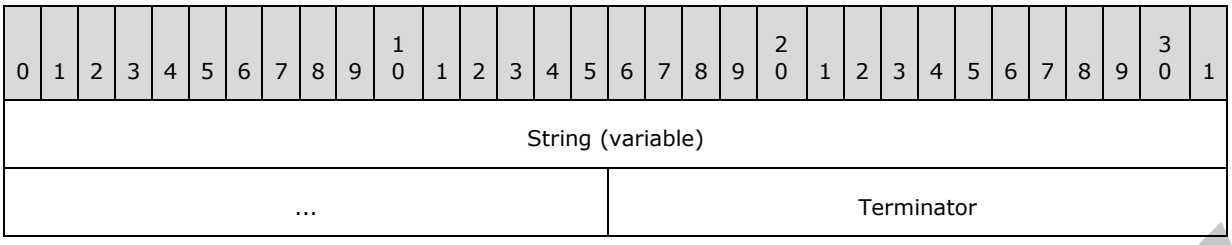
Buffer (variable): This field MUST exist only when the property type is **PtypBinary**. The **Buffer** field contains an arbitrary binary stream (2). The size, in bytes, of the stream (2) is specified in the **Data** field.

2.2.6.2 PidTagViewDescriptorStrings Property

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

The client MUST store the display strings referenced in the **PidTagViewDescriptorBinary** property (section 2.2.6.1) separately in the **PidTagViewDescriptorStrings** property ([MS-OXPROPS] section 2.1045).<18> The client MUST concatenate the strings in the same order in which the strings are referenced in the **PidTagViewDescriptorBinary** property. The first set of strings consists of the **display names** of each of the **ColumnInfo** structures, followed by the value of each restriction structure's **PropValue** field that uses the **PtypString** property type.

The client MUST use the following binary layout in the **PidTagViewDescriptorStrings** property.



String (variable): This field is an arbitrary length buffer that contains a string. Clients SHOULD format the string using the current code page of the client. The application MUST NOT include the byte value 0x0A00, which corresponds to the newline character, in the string.

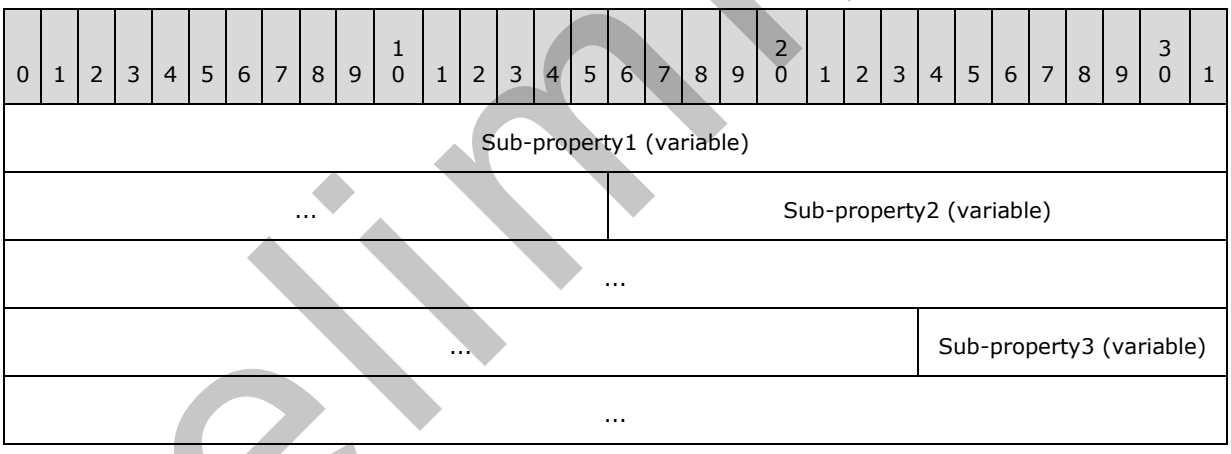
Terminator (2 bytes): This field contains the value 0x0A00. The application MUST include a **Terminator** field after every **String** field, including the last **String** field in the stream (2).

2.2.7 Folder Flags

Folder flags consist of a collection of small properties packed into a single binary property on a folder. The primary purpose of the folder flags is to store Boolean flags that affect the folder's display options.

The folder flags can also be used to store additional properties, such as a unique identifier for the folder that can be used to associate it with a specific feature or with a description of that folder that has been saved elsewhere.

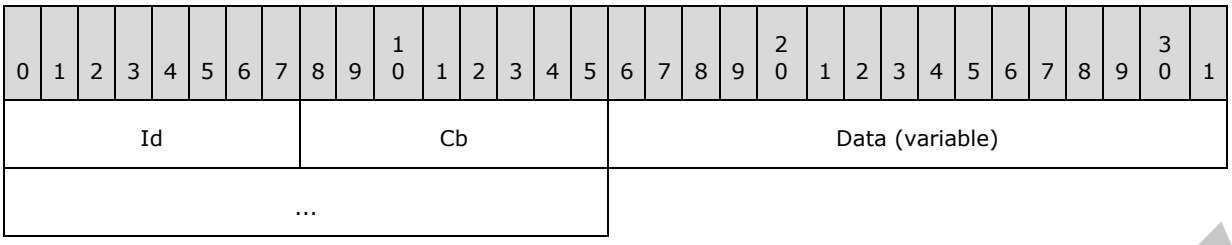
The **PidTagExtendedFolderFlags** property ([MS-OXOSRCH] section 2.2.2.1.2) can be set on a folder. If the property is set, the value of this property MUST be a binary stream (2) that contains encoded **subproperties** for the folder. The format of the binary stream (2) MUST be as follows.



The binary stream (2) is divided into variable-length subproperty fields. The subproperty fields are byte-aligned within the binary stream (2). Each subproperty MUST be encoded as specified in section 2.2.7.1.

2.2.7.1 Sub-property

The sub-property fields for folder flags specified in section 2.2.7.1 have the following format.



Id (1 byte): The subproperty ID value. The value of this field SHOULD be one of the following. All other values of the **Id** field are reserved and MUST be ignored by the application. If the application needs to rewrite the **PidTagExtendedFolderFlags** property ([MS-OXOSRCH] section 2.2.2.1.2) with different values for the subproperties that it does understand, it MUST preserve the values of any subproperties that it did not understand. Each valid subproperty ID MUST appear 0 to 1 times in the **PidTagExtendedFolderFlags** property. The subproperties can appear in any order within the the **PidTagExtendedFolderFlags** property stream (2).

Flag name	Value	Data format
Invalid	0x00	As specified in section 2.2.7.1.1.
ExtendedFlags	0x01	As specified in section 2.2.7.1.2.
SearchFolderID	0x02	As specified in section 2.2.7.1.3.
SearchFolderTag	0x03	As specified in [MS-OXOSRCH].
Reserved	0x04	N/A
ToDoFolderVersion	0x05	As specified in section 2.2.7.1.4.
Reserved	0x06	N/A

Cb (1 byte): This field specifies the unsigned size, in bytes, of the **Data** field of the subproperty.

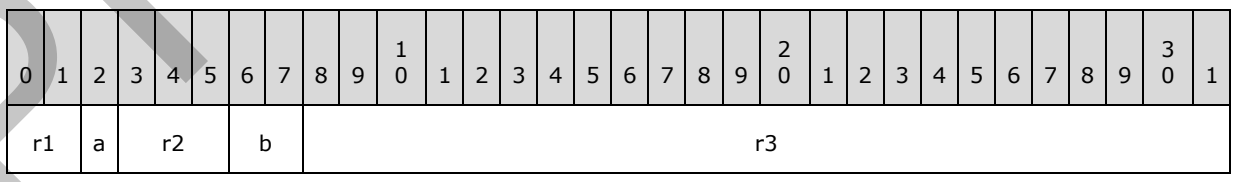
Data (variable): This field contains the value of the subproperty. This field MUST be a variable-length buffer. Because the size is specified in a single unsigned byte in the **Cb** field, the minimum size of the buffer is 0 bytes and the maximum size is 255 bytes. The interpretation of this field is specified in the table earlier in this section.

2.2.7.1.1 Invalid

If the **Id** field is set to **Invalid**, the value of the **Data** field is invalid. The application MUST NOT use it.

2.2.7.1.2 ExtendedFlags

If the **Id** field is set to **ExtendedFlags**, the value of the **Data** field is in the following format. If the subproperty does not exist, or if the **PidTagExtendedFolderFlags** property ([MS-OXOSRCH] section 2.2.2.1.2) is not set on the folder, each flag SHOULD assume the specified default value.



r1 (2 bits): Reserved. The application can set these flags to any value when writing the subproperty. The application MUST ignore these flags when reading the subproperty, but it MUST preserve preexisting values if it rewrites the subproperty.

a (1 bit): If the folder is subject to an administrative retention policy, this flag controls whether the application displays a string that describes the policy. The possible values are listed in the following table.

Value	Meaning
0	The application SHOULD<19> display a policy description. This is the default value.
1	The application MUST NOT display a policy description.

r2 (3 bits): Reserved. The application can set these flags to any value when writing the subproperty. The application MUST ignore these flags when reading the subproperty, but it MUST preserve preexisting values if it rewrites the subproperty.

b (2 bits): These 2 bits control whether the application SHOULD display the total number of messages in the folder or only the number of unread messages in the folder. The possible values are listed in the following table.

Value	Meaning
00	The application uses the default value for this folder.
01	The application uses the number of unread messages in the folder. This is the default value for all folders except for the Outbox, Drafts, and Junk E-mail special folders, as specified in [MS-OXOSFLD].
10	The application uses the total number of messages in the folder. This is the default value for the Outbox, Drafts, and Junk E-mail special folders.
11	This value is invalid. The application MUST NOT use it.

r3 (3 bytes): Reserved. The application can set these flags to any value when writing the subproperty. The application MUST ignore these flags when reading the subproperty, but it MUST preserve preexisting values if it rewrites the subproperty.

2.2.7.1.3 SearchFolderID

If the **Id** field is set to **SearchFolderID**, the value of the **Data** field is a 16-byte field. When the application creates a persistent **search folder (2)**, as specified in [MS-OXOSRCH] section 2.2.1, it MUST set this field on the folder to the same value as the **PidTagSearchFolderId** property ([MS-OXOSRCH] section 2.2.1.2.1) on the message.

2.2.7.1.4 ToDoFolderVersion

If the **Id** field is set to **ToDoFolderVersion**, the value of the **Data** field is a 4-byte field. When the application creates the To-Do Search folder as specified in [MS-OXOSFLD] section 3.1.4.1.2, it MUST be a bitmask that indicates which stream (2) properties exist on the message. The stream (2) types, and thus the flags, are not mutually exclusive, which corresponds to the little-endian integer value of 0x000C0000.

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

2.2.8 Conversation Actions

Conversation actions are a limited set of **actions (3)** that a user applies to all E-mail objects, currently in the **message store** or delivered in the future, that share the same value of the **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2) and thus are part of the same conversation. Unlike **rules (1)**, as specified in [MS-OXORULE], at most one conversation action applies to any given E-mail object, and conversation actions can be automatically deleted after a certain period of disuse.

A conversation action is a combination of any of the following actions (3):

- Ignore
- Stop Ignoring
- Move to Folder
- Move to Store
- Stop Move
- Add Category
- Remove Category
- Clear All Categories

The client and server SHOULD<20> persist conversation action settings in an FAI message in the conversation actions settings special folder, as specified in [MS-OXOSFLD] section 2.2.5.1. The relevant properties of a conversation action FAI message are enumerated in sections 2.2.8.1 through 2.2.8.10.

2.2.8.1 PidLidConversationActionLastAppliedTime Property

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

The **PidLidConversationActionLastAppliedTime** property ([MS-OXPROPS] section 2.82) SHOULD<21> be the time (in UTC) that an E-mail object was last received in the conversation or the last time that the user modified the conversation action, whichever occurs later.

2.2.8.2 PidLidConversationActionMaxDeliveryTime Property

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

The **PidLidConversationActionMaxDeliveryTime** property ([MS-OXPROPS] section 2.83) SHOULD<22> be the maximum value of the **PidTagMessageDeliveryTime** property ([MS-OXOMSG] section 2.2.3.9) of all the E-mail objects modified in response to the last time the user changed a conversation action on the client. If no E-mail objects were changed, this property SHOULD be set to 00:00:00 Apr 1, 1601 (UTC).

2.2.8.3 PidLidConversationActionMoveFolderEid Property

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

The **PidLidConversationActionMoveFolderEid** property ([MS-OXPROPS] section 2.84) SHOULD<23> be set based on the action (3) or actions (3) being performed on the conversation, as shown in the following table.

Action on the conversation	Value
Move to Folder (move to a folder in the same message store as this FAI message)	The **EntryID** of that folder
Move to Store (move to a folder in a different message store than this FAI message)	The EntryID of that folder
Ignore	Byte array of size zero
None of the above	Not set

2.2.8.4 PidLidConversationActionMoveStoreEid Property

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

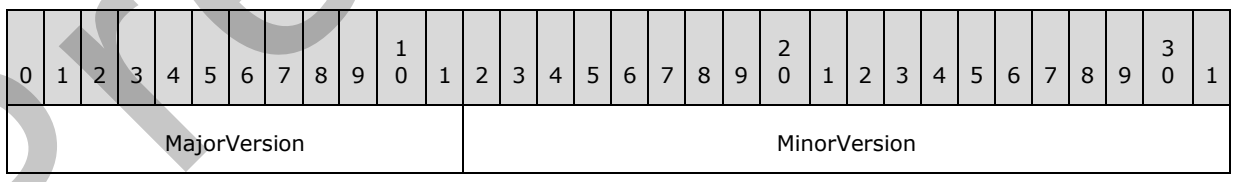
The **PidLidConversationActionMoveStoreEid** property ([MS-OXPROPS] section 2.85) SHOULD<24> be set based on the action (3) or actions (3) being performed on the conversation, as shown in the following table.

Action on the conversation	Value
Move to Folder (move to a folder in the same message store as this FAI message)	Not set
Move to Store (move to a folder in a different message store than this FAI message)	The EntryID of that message store
Ignore	Not set
None of the above	Not set

2.2.8.5 PidLidConversationActionVersion Property

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

The **PidLidConversationActionVersion** property ([MS-OXPROPS] section 2.86) SHOULD<25> be set to the version of the conversation action FAI message, as specified in section 2.2.8. The format of the value of this property MUST be as follows.



MajorVersion (12 bits): This field indicates the version compatibility of the conversation action FAI message. It MUST be set to 0x004.

MinorVersion (20 bits): This value is implementation-specific and can be set to any value. It SHOULD indicate the version of the client or server that last modified the conversation action FAI message.

2.2.8.6 PidLidConversationProcessed Property

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

The **PidLidConversationProcessed** property ([MS-OXPROPS] section 2.87) specifies a sequential number to be used in the processing of a **conversation action**, as specified in section 3.1.5.1.

2.2.8.7 PidNameKeywords Property

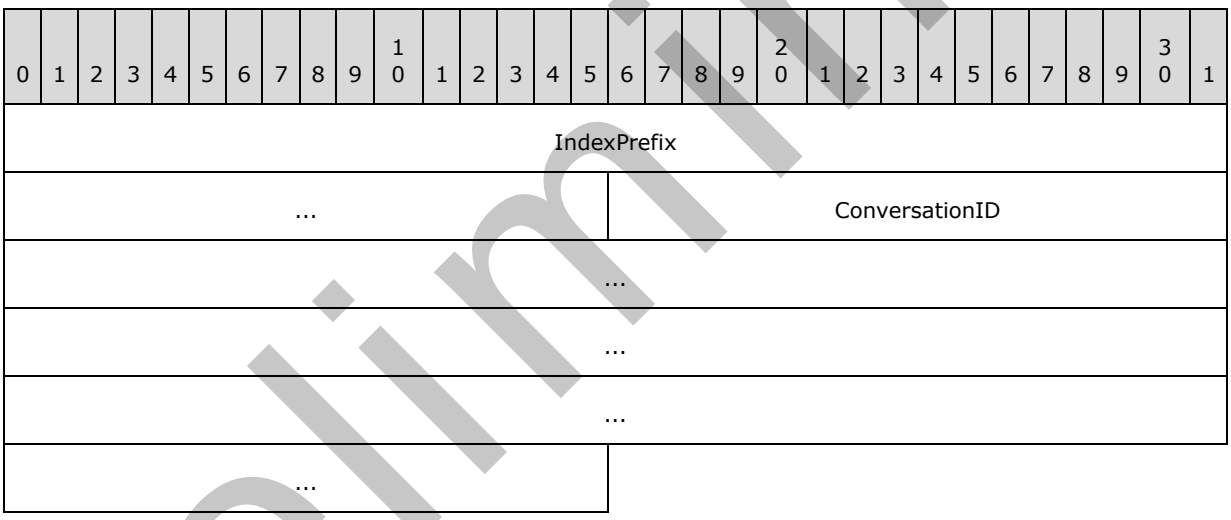
Type: **PtypMultipleString** ([MS-OXCDATA] section 2.11.1)

If this conversation is being categorized, the **PidNameKeywords** property ([MS-OXPROPS] section 2.444) SHOULD<26> be set to the list of categories (3) that are being set on incoming E-mail objects in the conversation. Otherwise, this property SHOULD NOT be set.

2.2.8.8 PidTagConversationIndex Property

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

The **PidTagConversationIndex** property ([MS-OXPROPS] section 2.641) SHOULD<27> be set to a 22-byte array as specified below.



IndexPrefix (6 bytes): This field MUST be set to 0x010000000000.

ConversationID (16 bytes): This field MUST be set to the value of the **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2) shared by all the E-mail objects in the conversation.

2.2.8.9 PidTagMessageClass Property

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

The **PidTagMessageClass** property ([MS-OXPROPS] section 2.776) SHOULD<28> be set to "IPM.ConversationAction".

2.2.8.10 PidTagSubject Property

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

The **PidTagSubject** property ([MS-OXPROPS] section 2.1021) SHOULD<29> be set to "Conv.Action: " followed by the value of the **PidTagNormalizedSubject** property (section 2.2.9.2) of an E-mail object in the conversation. This correlates the FAI message to the E-mail objects in the conversation.

2.2.9 Navigation Shortcuts

Navigation shortcuts are objects that contain identifying information to locate a folder in a message store. Clients can use navigation shortcuts to provide quick access to particular folders via the client's user interface.

Navigation shortcuts are stored as FAI messages, as specified in [MS-OXCMSG], in the **Common Views folder**, as specified in [MS-OXOSFLD], within a message database.

These messages possess additional properties that describe navigation shortcuts. These properties are described in sections 2.2.9.1 through 2.2.9.19.<30>

2.2.9.1 PidTagMessageClass Property

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

The **PidTagMessageClass** property ([MS-OXPROPS] section 2.776) identifies the message as a navigation shortcut message, as specified in section 2.2.9. The value is "IPM.Microsoft.WunderBar.Link".

2.2.9.2 PidTagNormalizedSubject Property

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

The **PidTagNormalizedSubject** property ([MS-OXPROPS] section 2.801) specifies the display name of the navigation shortcut, as specified in section 2.2.9.

2.2.9.3 PidTagWlinkGroupHeaderID Property

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

The **PidTagWlinkGroupHeaderID** property ([MS-OXPROPS] section 2.1059) specifies the ID of the navigation shortcut, as specified in section 2.2.9, that groups other navigation shortcuts. This property SHOULD be set only when the value of the **PidTagWlinkType** property (section 2.2.9.5) is **wblHeader**.

2.2.9.4 PidTagWlinkSaveStamp Property

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

The **PidTagWlinkSaveStamp** property ([MS-OXPROPS] section 2.1064) specifies an integer that allows a client to identify with a high probability whether the navigation shortcut, as specified in section 2.2.9, was saved by the current client session. When writing a navigation shortcut, the client writes the Session ID, as specified in section 3.1.3.1, as the value for this property.

2.2.9.5 PidTagWlinkType Property

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

The **PidTagWlinkType** property ([MS-OXPROPS] section 2.1067) specifies the type of navigation shortcut, as specified in section 2.2.9. The valid values of this property are as follows.

Value	Value name	Meaning
0x00000000	wblNormalFolder	The shortcut points to a folder other than a search folder (2), **group header** shortcut, or a folder owned by a different user.
0x00000001	wblSearchFolder	The shortcut points to a search folder (2).
0x00000002	wblSharedFolder	The shortcut points to a folder that is owned by a different user.
0x00000004	wblHeader	The shortcut points to the group header shortcut.

2.2.9.6 PidTagWlinkFlags Property

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

The **PidTagWlinkFlags** property ([MS-OXPROPS] section 2.1056) is a bit field in which each bit SHOULD be set to 1 if the associated condition in the following table applies and to 0 otherwise. The possible bit values are as follows.

Value	Flag name	Meaning
0x00000001	**sipPublicFolder**	Shortcut points to a server **public folder**.
0x00000004	**sipImapFolder**	Shortcut points to a folder that is accessed via the **Internet Message Access Protocol - Version 4 (IMAP4)**.
0x00000008	**sipWebDavFolder**	Shortcut points to a folder that is accessed via the **Web Distributed Authoring and Versioning Protocol (WebDAV)**.
0x00000010	**sipSharePointFolder**	Shortcut points to a folder that is accessed via SharePoint Products and Technologies.
0x00000020	**sipRootFolder**	Shortcut points to a **Root folder** (**interpersonal messaging subtree**).
0x00000100	**sipSharedOut**	Shortcut points to a folder that has been shared with other users. This value can be used to display a visual indicator and does not set any **permissions**.
0x00000200	**sipSharedIn**	Shortcut points to a folder that is owned by another user. This value can be used to display a visual indicator and does not set any permissions.
0x00000400	**sipPersonFolder**	Shortcut points to a folder that belongs to another user.
0x00000800	**sipiCal**	Shortcut points to a folder that accesses data from an iCalendar data source (see [MS-OXCICAL]).
0x00001000	**sipOverlay**	Shortcut points to a **Calendar folder**. The contents of the Calendar folder, if displayed with other **calendars**, are merged into one calendar and the ordering determined by the **PidTagWlinkOrdinal** property (section 2.2.9.7). Merging is

Value	Flag name	Meaning
		purely a display; no objects are created or moved between folders.
0x00002000	**sipOneOffName**	Shortcut has been renamed such that the value of the **PidTagNormalizedSubject** property (section 2.2.9.2) of the shortcut does not match the value of the **PidTagNormalizedSubject** property of the folder, or the shortcut is a group header and has been renamed from the default value.

All bits not specified in the above table are reserved. They MUST be ignored, but if set, they are to be preserved.

2.2.9.7 PidTagWlinkOrdinal Property

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

The **PidTagWlinkOrdinal** property ([MS-OXPROPS] section 2.1061) specifies a variable-length binary property that SHOULD be used to sort shortcuts lexicographically. For example, to insert a shortcut C between shortcut A with the one-byte ordinal value of 128 and shortcut B with the one-byte ordinal value of 129, shortcut C can be assigned the two-byte ordinal 128, 128. The final byte of this property MUST NOT be 0 or 255 to ensure shortcuts can always be inserted before and after other shortcuts.

2.2.9.8 PidTagWlinkEntryId Property

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

The **PidTagWlinkEntryId** property ([MS-OXPROPS] section 2.1055) specifies the EntryID of the folder pointed to by the shortcut.

2.2.9.9 PidTagWlinkRecordKey Property

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

The **PidTagWlinkRecordKey** property ([MS-OXPROPS] section 2.1062) specifies the value of the **PidTagRecordKey** property ([MS-OXCPRPT] section 2.2.1.8) of the folder pointed to by the shortcut.

2.2.9.10 PidTagWlinkStoreEntryId Property

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

The **PidTagWlinkStoreEntryId** property ([MS-OXPROPS] section 2.1066) specifies the value of the **PidTagStoreEntryId** property ([MS-OXPROPS] section 2.1016) of the folder pointed to by the shortcut.

2.2.9.11 PidTagWlinkFolderType Property

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

The **PidTagWlinkFolderType** property ([MS-OXPROPS] section 2.1057) specifies the type of folder pointed to by the shortcut. The possible values are listed in the following table.

Value	Value name	Meaning
{0x00780600, 0x0000, 0x0000, {0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46}}	**CLSID_MailFolder**	The folder is a mail folder.
{0x02780600, 0x0000, 0x0000, {0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46}}	**CLSID_CalendarFolder**	The folder is a Calendar folder.
{0x01780600, 0x0000, 0x0000, {0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46}}	**CLSID_ContactFolder**	The folder is a Contact folder.
{0x03780600, 0x0000, 0x0000, {0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46}}	**CLSID_TaskFolder**	The folder is a Task folder.
{0x04780600, 0x0000, 0x0000, {0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46}}	**CLSID_NoteFolder**	The folder is a Note folder.
{0x08780600, 0x0000, 0x0000, {0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x 00, 0x00, 0x46}}	**CLSID_JournalFolder**	The folder is a Journal folder.

2.2.9.12 PidTagWlinkGroupClsid Property

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

The **PidTagWlinkGroupClsid** property ([MS-OXPROPS] section 2.1058) specifies the value of the **PidTagWlinkGroupHeaderID** property (section 2.2.9.3) of the group header associated with the shortcut.

2.2.9.13 PidTagWlinkGroupName Property

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

The **PidTagWlinkGroupName** property ([MS-OXPROPS] section 2.1060) specifies the value of the **PidTagNormalizedSubject** property (section 2.2.9.2) of the group header associated with the shortcut.

2.2.9.14 PidTagWlinkSection Property

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

The **PidTagWlinkSection** property ([MS-OXPROPS] section 2.1065) specifies the section where the shortcut can be grouped. The possible values are listed in the following table.

Value	Value name	Meaning
0x00000001	**wbsidMailFavorites**	Shortcut is grouped under Mail.
0x00000002	NA	NA
0x00000003	**wbsidCalendar**	Shortcut is grouped under Calendar.
0x00000004	**wbsidContacts**	Shortcut is grouped under Contacts.
0x00000005	**wbsidTasks**	Shortcut is grouped under Tasks.
0x00000006	**wbsidNotes**	Shortcut is grouped under Notes.
0x00000007	**wbsidJournal**	Shortcut is grouped under Journal.

2.2.9.15 PidTagWlinkCalendarColor Property

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

The **PidTagWlinkCalendarColor** property ([MS-OXPROPS] section 2.1053) specifies the background color of the calendar. The RGB values listed in the following table can be used.

Value	Color (R, G, B)
-1	Automatic (determined by implementation)
0	141, 174, 217
1	156, 191, 139
2	209, 149, 170
3	176, 182, 190
4	176, 182, 190
5	140, 140, 215
6	141, 193, 157
7	211, 150, 150
8	186, 186, 137
9	174, 153, 216
10	195, 176, 141
11	139, 191, 174
12	144, 182, 200
13	255, 223, 134
14	150, 169, 209

2.2.9.16 PidTagWlinkAddressBookEID Property

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

The **PidTagWlinkAddressBookEID** property ([MS-OXPROPS] section 2.1051) specifies the value of the **PidTagEntryId** property ([MS-OXCPERM] section 2.2.4) of the user that the folder belongs to, as specified in [MS-OXOABK] section 2.2.3.2. This property SHOULD<31> be set on calendar shortcuts.

2.2.9.17 PidTagWlinkAddressBookStoreEID Property

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

The **PidTagWlinkAddressBookStoreEID** property ([MS-OXPROPS] section 2.1052) specifies the value of the **PidTagStoreEntryId** property ([MS-OXPROPS] section 2.1016) of the current user (not the owner of the folder). This property SHOULD<32> be set on calendar shortcuts.

2.2.9.18 PidTagWlinkClientID Property

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

The **PidTagWlinkClientID** property ([MS-OXPROPS] section 2.1054) specifies the Client ID that allows the client to determine whether the shortcut was created on the current machine or by the currently logged-in user via an equality test. The ID is specified in section 3.1.3.1. If this property is set, the client SHOULD<33> compare the value of this property to the Client ID and display the shortcut only if the **GUIDs** match.

2.2.9.19 PidTagWlinkROGroupType Property

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

The **PidTagWlinkROGroupType** property ([MS-OXPROPS] section 2.1063) specifies the type of group header. If the property does not exist, the client SHOULD<34> assume a value of -1. The possible values are listed in the following table.

Value	Value name	Meaning
-1	wbrogUndefined	None.
0	wbrogMyDepartment	Group contains shortcuts to users in his department.
1	wbrogOtherDepartment	Group contains shortcuts to users in another department.
2	wbrogDirectReportGroup	Group contains shortcuts to users in his direct reporting group.
3	wbrogCoworkerGroup	Group contains shortcuts to coworkers of the user.
4	wbrogDL	Group contains shortcuts to members of a **distribution list**.

3 Protocol Details

3.1 Client Details

3.1.1 Abstract Data Model

None.

3.1.2 Timers

None.

3.1.3 Initialization

3.1.3.1 Navigation Shortcuts

On initialization of the client, the client generates a Session ID that is valid as long as the client is running. The Session ID is of type **PtypInteger32** and is a randomly generated value. This value is saved as the value of the **PidTagWlinkSaveStamp** property (section 2.2.9.4).

The client also generates a Client ID for the user that is valid across sessions on first-run initialization of the client. The Client ID is of type **PtypBinary** and is a randomly generated value. This value is saved as the value of the **PidTagWlinkClientID** property (section 2.2.9.18).

Navigation shortcuts are specified in section 2.2.9.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Reading Configuration Data

To read settings in a configuration data settings group, as specified in section 2.2.2, the client MUST open the special folder that contains the configuration data message. The client MUST call the **RopGetContentsTable** ROP ([MS-OXCROPS] section 2.2.4.14) with the **Associated** flag to open the FAI contents table, as specified in [MS-OXCFOLD] section 2.2.1.14.

The client MUST find all the rows in the FAI contents table that have the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) specified by the configuration data message that the client is trying to open, by using steps equivalent to the following, as specified in [MS-OXCTABL]:

- Send the **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1) with the following properties:
  - **PidTagFolderId** ([MS-OXCFOLD] section 2.2.2.2.1.5)
  - **PidTagMid** ([MS-OXCFXICS] section 2.2.1.2.1)
  - **PidTagMessageClass**
  - **PidTagRoamingDatatypes** (section 2.2.2.1)
- Send the **RopSortTable** ROP ([MS-OXCROPS] section 2.2.5.2) with a sort order (3) that includes the following properties:
  - **PidTagMessageClass**, followed by

▪**PidTagLastModificationTime** ([MS-OXCMSG] section 2.2.2.2)

- Send the **RopFindRow** ROP ([MS-OXCROPS] section 2.2.5.13), searching for a match on the **PidTagMessageClass** property.
- Send the **RopQueryRows** ROP ([MS-OXCROPS] section 2.2.5.4) repeatedly until either the end of the table or a row with a **PidTagMessageClass** property that no longer matches the configuration data message is encountered.
- Based on the subsort by the **PidTagLastModificationTime** property, pick the message with the most recent (the greatest value) modification time that includes the bit that matches the **PidTagRoamingDatatypes** property specified by the configuration data message. If none of the messages match the **PidTagRoamingDatatypes** property of the configuration data message, the client MUST pick the most recently modified of all the messages.

If the client cannot find a row that matches the **PidTagMessageClass** and **PidTagRoamingDatatypes** properties of the configuration data message, that group of settings does not exist. When reading the settings, the client MUST use default values for those settings when the configuration data message cannot be found.

If the client found a matching row, it MUST open the existing message as specified in [MS-OXCMSG] section 2.2.3.1 by opening the message, using the **PidTagFolderId** and **PidTagMid** properties from the table row and setting the **ReadWrite** flag in the **OpenModeFlags** field, by sending the **RopOpenMessage** ROP ([MS-OXCROPS] section 2.2.6.1).

If the client found a matching message, it MUST retrieve the serialized settings stream (2) from the property specified by the configuration data message by using steps equivalent to the following, as specified in [MS-OXCPRPT] section 2.2.14:

1. Open a Stream object **handle** on the stream (2) property specified by the configuration data message by sending the **RopOpenStream** ROP ([MS-OXCROPS] section 2.2.9.1).
2. Read the serialized settings by using the **RopReadStream** ROP ([MS-OXCROPS] section 2.2.9.2).

If multiple configuration data messages of the same type are found, the configuration data messages are deemed in conflict and MUST be resolved. If no specific conflict resolution algorithm is available, the first matching message SHOULD be used, and the client SHOULD delete the rest of the matching configuration data messages from the message store. Regardless of the method of resolution, a client SHOULD resolve such conflicts as soon as possible and SHOULD delete any duplicates, leaving only one configuration data message that is the result of the conflict resolution.

3.1.4.1.1 Reading Dictionaries

The client MUST prepopulate the dictionaries with default name-value property pairs, as specified in section 2.2.5.1.

The client MUST read any existing settings from the configuration data message, as specified in section 3.1.4.1. If any existing settings are found, the client MUST parse the XML document, as specified in section 2.2.5.1.

If the XML document does exist, and the XML document includes one of the valid **OLPrefsVersion** attributes specified in section 2.2.5.1, the client MUST then set the name-value pairs on the dictionary, overriding any default values that were prepopulated in the dictionary with matching names.

If the XML document did not exist, a valid **OLPrefsVersion** attribute did not exist or was incorrect, or any default settings did not overlap with previously saved settings, or if the client changes a setting after reading them, the client MUST write the contents of the dictionary to the configuration data message, as specified in section 3.1.4.1.

3.1.4.1.2 Reading Working Hours

The client MUST read any existing settings from the configuration data message, as specified in section 3.1.4.1. If any existing settings are found, the client MUST parse the XML document, as specified in section 2.2.5.2.

If the client could not find a matching configuration data message and it used default values,<35> or if the user changes the preferred working hours through the client UI, the client MUST generate the XML document as specified in section 2.2.5.2 and save it to the configuration data message as specified in section 3.1.4.2

When viewing the contents of another user's Calendar folders or displaying their **free/busy status** data, the client MUST attempt to open the other user's working hours configuration data message and translate the settings from the other user's time zone to the time zone of the client. The client MUST use the other user's preferred working hours in place of the client's settings when displaying the other user's Calendar folders.

If the client is unable to read the configuration data message from the other user's Calendar special folder (because the other user's message store is inaccessible or the client has not been granted sufficient permissions to access the special folder), the client MUST treat all times as being within the other user's preferred working hours.

3.1.4.1.3 Reading Category List

A category list configuration data message, as specified in in section 2.2.2, is saved as an FAI message in the user's default Calendar folder. A client can read the category list, as specified in section 2.2.5.2.2, at any time.

When clients encounter unknown tags or attributes, they SHOULD ignore them, but they SHOULD also rewrite them as-is when they rewrite the category list back to the configuration data message.

All times are to be stored relative to UTC. When a category (3) is applied to a message, as specified in section 3.1.4.7, or the user-visible properties of the category (3) are changed (such as color or shortcut key), the client SHOULD update the **lastTimeUsed** attribute and, depending on whether the client separates different message types (Mail, Calendar, Contacts, Tasks, Notes, Journal), SHOULD update the appropriate time stamp attribute as specified in section 2.2.5.2.2 with the current time.

When viewing the contents of another user's folders, the client MUST try to open the other user's category list configuration data message. If the client is able to read the configuration data message from the other user's Calendar special folder, the client MUST use the other user's category list settings, including color assignments, in place of the client's settings when it displays the other user's folders.

If the client is unable to read the configuration data message from the other user's Calendar special folder (because the other user's message store is inaccessible or the client has not been granted sufficient permissions to access the special folder), the client MUST fall back to its own category list settings.

3.1.4.2 Writing Configuration Data

To write settings in a configuration data settings group, as specified in section 2.2.2, the client MUST first look for a preexisting configuration data message with preexisting settings, as specified in section 3.1.4.1.

If the client found a matching message or created a new one, it MUST retrieve the serialized settings stream (2) from the property specified by the configuration data message by using steps equivalent to the following, as specified in [MS-OXCPRPT] section 2.2.15:

1. Open a Stream object handle on the stream (2) property specified by the configuration data message by sending the **RopOpenStream** ROP ([MS-OXCROPS] section 2.2.9.1).

2. Read the serialized settings by using the **RopReadStream** ROP ([MS-OXCROPS] section 2.2.9.2).

If the message does not exist, the client MUST create the message, as specified in [MS-OXCMSG] section 2.2.3.2, by sending the **RopCreateMessage** ROP ([MS-OXCROPS] section 2.2.6.2) on the folder and passing the **Associated** flag.

If the client found a matching message or created a new one, it MUST save the serialized settings stream (2) into the property specified by the configuration data message by using steps equivalent to the following, as specified in [MS-OXCPRPT] section 2.2.16:

1. Open a Stream object handle on the stream (2) property specified by the configuration data message by sending the **RopOpenStream** ROP.

2. Write the serialized settings by using the **RopWriteStream** ROP ([MS-OXCROPS] section 2.2.9.3).

3. Persist the stream (2) back to the property by sending the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

4. Persist changes to the message by sending the **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3).

3.1.4.2.1 Writing Dictionaries

The client MUST read any existing settings from the configuration data message, as specified in section 3.1.4.1. If any existing settings are found, the client MUST parse the XML document, as specified in section 2.2.5.1.

The client MUST write the contents of the dictionary to the configuration data message, as specified in section 3.1.4.2.

3.1.4.2.2 Writing Working Hours

The client MUST read any existing settings from the configuration data message, as specified in section 3.1.4.1. If any existing settings are found, the client MUST parse the XML document, as specified in section 2.2.5.2.

The client MUST generate the XML document as specified in section 2.2.5.2 and save it to the configuration data message as specified in section 3.1.4.2.

3.1.4.2.3 Writing Category List

The client MUST read any existing settings from the configuration data message, as specified in section 3.1.4.1. If any existing settings are found, the client MUST parse the XML document, as specified in section 2.2.5.2.

When clients encounter unknown tags or attributes, they SHOULD ignore them, but they SHOULD also rewrite them as-is when they rewrite the category list back to the configuration data message.

The client MUST generate the XML document as specified in section 2.2.5.2 and save it to the configuration data message as specified in section 3.1.4.2.

Each category (3) in the category list, as specified in section 2.2.5.2.2, contains the following attributes:

- **lastTimeUsed**
- **lastTimeUsedMail**
- **lastTimeUsedCalendar**
- **lastTimeUsedContacts**
- **lastTimeUsedTasks**
- **lastTimeUsedNotes**
- **lastTimeUsedJournal**

The **lastTimeUsed** attribute MUST be set on all categories (3). All others are optional and depend on whether the client shows different message types in separate windows or panes.

All times MUST be stored relative to UTC. When a category (3) is applied to a message, as specified in section 3.1.4.7, or the user visible properties of the category (3) are changed (such as color or shortcut key), the client SHOULD update the **lastTimeUsed** attribute and, depending on whether the client separates different message types, SHOULD replace the appropriate type-specific time stamp with the current time.

3.1.4.3 Reading View Definitions

To read the list of available view definitions for a folder, as specified in section 2.2.6, the client MUST enumerate all of the view definition FAI messages in the folders, searching for a match on the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) and the **PidTagViewDescriptorVersion** property ([MS-OXPROPS] section 2.1046) as specified in section 2.2.6.

After the client has built the list of view definition messages, it can select one of them by using the **PidTagViewDescriptorName** property ([MS-OXPROPS] section 2.1044). After it has selected a view definition message, the client MUST read the settings from the **PidTagViewDescriptorBinary** property (section 2.2.6.1) and the **PidTagViewDescriptorStrings** property (section 2.2.6.2) on the message.

3.1.4.4 Writing View Definitions

To write settings in a view definition, as specified in section 2.2.6, the client MUST open the folder that contains the view definition message. The client MUST save the view definition message in the folder that will display that view.

The client MUST call the **RopGetContentsTable** ROP ([MS-OXCROPS] section 2.2.4.14) with the **Associated** flag to open the FAI contents table, as specified in [MS-OXCFOLD] section 2.2.1.14.

If a view definition message already exists with the same **PidTagViewDescriptorName** property ([MS-OXPROPS] section 2.1044) in the same folder, the client MUST open that message and save the view definition there. The client MUST search for a matching row in the FAI contents table by using steps equivalent to the following, as specified in [MS-OXCTABL] section 2.2.2:

1. Send the **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1) with the following properties:

   - **PidTagFolderId** ([MS-OXCFOLD] section 2.2.2.2.1.5)
   - **PidTagMid** ([MS-OXPROPS] section 2.790)
   - **PidTagMessageClass** ([MS-OXCMSG] section 2.2.1.3)
   - **PidTagViewDescriptorVersion** ([MS-OXPROPS] section 2.1046)
   - **PidTagViewDescriptorName** ([MS-OXPROPS] section 2.1044)

2. Send the **RopSortTable** ROP ([MS-OXCROPS] section 2.2.5.2) with a sort order (3) that includes the following properties:

   - **PidTagMessageClass**, followed by
   - **PidTagViewDescriptorVersion**
   - **PidTagViewDescriptorName**

3. Send the **RopFindRow** ROP ([MS-OXCROPS] section 2.2.5.13), searching for a match on the **PidTagMessageClass**, **PidTagViewDescriptorVersion**, and **PidTagViewDescriptorName** properties, as specified in section 2.2.6.

4. Send the **RopQueryRows** ROP ([MS-OXCROPS] section 2.2.5.4) to retrieve a single row and get the **PidTagFolderId** and **PidTagMid** properties of the matching message from the row.

If the message does not exist, the client MUST create the message as specified in [MS-OXCMSG] section 2.2.3.2 by sending the **RopCreateMessage** ROP ([MS-OXCROPS] section 2.2.6.2) on the folder, and passing the **Associated** flag.

If the client found a matching row, it MUST send the **RopOpenMessage** ROP ([MS-OXCROPS] section 2.2.6.1) to open the message, using the **PidTagFolderId** and **PidTagMid** properties from the table row and setting the **ReadWrite** flag in the **OpenModeFlags** field, as specified in [MS-OXCMSG] section 2.2.3.1.

If the client found a matching message or created a new one, it MUST save the serialized settings streams (2) on the properties specified in section 2.2.6 by using steps equivalent to the following, as specified in [MS-OXCPRPT] section 2.2.16:

1. Send the **RopOpenStream** ROP ([MS-OXCROPS] section 2.2.9.1) to open Stream object handles on the **PidTagViewDescriptorBinary** property (section 2.2.6.1) and the **PidTagViewDescriptorStrings** property (section 2.2.6.2).

2. Write the serialized settings by using the **RopWriteStream** ROP ([MS-OXCROPS] section 2.2.9.3).

3. Persist the stream (2) back to the property by sending the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

4. Persist changes to the message by sending the **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3).

3.1.4.5 Reading Folder Flags

To read folder flags on a folder, as specified in section 2.2.7, the client MUST obtain a handle to the folder using the **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1), as specified in [MS-OXCFOLD] section 3.1.4.1. The client MUST then retrieve the data for the folder flags by sending the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3) with the **PidTagExtendedFolderFlags** property ([MS-OXOSRCH] section 2.2.2.1.2) in the **PropertyTags** field of the request buffer ([MS-OXCROPS] section 2.2.8.3.1). The binary data MUST be interpreted as specified in section 2.2.7. Reading and writing each of the subproperties in the folder flags are triggered by different events.

3.1.4.5.1 Reading ExtendedFolderFlags

The client MUST read the bit flags in the **ExtendedFolderFlags** subproperty before it can display the folder in the UI.

3.1.4.5.2 Reading SearchFolderID

The client MUST read the value of the **SearchFolderID** subproperty from every search folder (2), as specified in [MS-OXOSRCH], in the Finder special folder. Any search folder (2) that has this subproperty is a persistent search folder (2), and the client SHOULD display the search folder (2) as such in the UI. Searches with the Finder special folder are further specified in [MS-OXOSRCH] section 3.1.4.1.2. For more details about the Finder special folder, see [MS-OXOSFLD].

3.1.4.5.3 Reading ToDoFolderVersion

The client MUST read the value of the **ToDoFolderVersion** subproperty from the To-Do Search folder before it displays the contents of that folder. If the To-Do Search folder does not exist, does not contain this subproperty, or does not contain the required value as defined in section 2.2.7, then the client MUST re-create the To-Do Search folder, as specified in [MS-OXOSFLD] section 3.1.4.1.2, or reset the criteria of the search folder (2).

3.1.4.6 Writing Folder Flags

To write folder flags on a folder, as specified in section 2.2.7, the client MUST obtain a handle to the folder using the **RopOpenFolder** ROP ([MS-OXCROPS] section 2.2.4.1), as specified in [MS-OXCFOLD] section 3.1.4.1. The client MUST then format the binary data as specified in section 2.2.7. The client MUST then write the data for the folder flags by sending the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) with the **PidTagExtendedFolderFlags** property ([MS-OXOSRCH] section 2.2.2.1.2) and the value in the **PropertyValues** field of the **RopSetProperties ROP request buffer** ([MS-OXCROPS] section 2.2.8.6.1). In each case where the client needs to write a new value of one of the subproperties to the folder, it MUST preserve the values of any other unmodified subproperties on the folder, as specified in section 2.2.7.

3.1.4.6.1 Writing ExtendedFolderFlags

Any time the user changes one of the display options for a folder, the client MUST rewrite the subproperty to the folder.

3.1.4.6.2 Writing SearchFolderID

The client MUST write the **SearchFolderID** subproperty on any new persistent search folders (2) that it creates.

3.1.4.6.3 Writing ToDoFolderVersion

When the client re-creates or resets the criteria on the To-Do Search folder, it MUST set the **ToDoFolderVersion** subproperty on the folder.

3.1.4.7 Applying a Category to a Message

A message can have a list of categories (3) stored in the **PidNameKeywords** property (section 2.2.8.7). To apply a new category (3) to a message, the client MUST read the current value of the **PidNameKeywords** property from the message and check to see whether the current value already contains the name of the new category (3). If the current value does not include the name of the new category (3), the client MUST insert the name of the category (3) in the list and set the new value of **PidNameKeywords** property on the message.

3.1.4.8 Performing a Conversation Action

Certain user actions performed on a conversation SHOULD<36> modify the conversation actions, as specified in section 2.2.8, on a conversation and trigger special handling logic:

- Ignore
- Stop Ignoring
- Move to Folder
- Stop Move
- Add Category
- Remove Category
- Clear All Categories

When the user performs one of these actions on a conversation, the client SHOULD identify all unique values of the **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2) on the E-mail objects and do the following for each **conversation ID**:

1. The client SHOULD locate the conversation action FAI message in the Conversation Actions Settings special folder with bytes 6-21 of the **PidTagConversationIndex** property (section 2.2.8.8) corresponding to the given conversation ID. If no such FAI message exists, proceed to step 4.
2. If the user is performing a Stop Ignoring action and the FAI message does not indicate that there is an Ignore conversation action (as specified in sections 2.2.8.3 and 2.2.8.4), the client SHOULD NOT perform the Stop Ignoring action for this conversation ID.
3. If the user is performing a Stop Move action and the FAI message does not indicate that there is a Move to Folder conversation action (as specified in sections 2.2.8.3 and 2.2.8.4), the client SHOULD NOT perform the Stop Move action for this conversation ID.
4. If no existing FAI message is found, a new FAI message SHOULD be created with the properties specified in section 2.2.8.

5. The client SHOULD<37> locate all E-mail objects on the message store with a **PidTagConversationId** property value matching the given conversation ID. If the operation cannot be done in a timely fashion, the client can proceed as though no E-mail objects were found.

6. The client SHOULD perform the appropriate action (3) on all located E-mail objects, as specified in the second column of the table following this procedure.

7. The client SHOULD set the latest value of the **PidTagMessageDeliveryTime** property ([MS-OXOMSG] section 2.2.3.9) from the located E-mail objects as the **PidLidConversationActionMaxDeliveryTime** property (section 2.2.8.2) on the FAI message. If no messages were found, the **PidLidConversationActionMaxDeliveryTime** property SHOULD be set to 00:00:00 (UTC) April 1, 1601.

8. The client SHOULD set the current time (in UTC) as the **PidLidConversationActionLastAppliedTime** property (section 2.2.8.1) on the FAI message.

9. The client SHOULD perform the appropriate action (3) on the FAI message, as specified in the third column of the table following this procedure.

10. If the server returned a version less than 14.0.324.0, as described in section 1.7, and the **PidLidConversationActionMoveFolderEid** (section 2.2.8.3) and **PidNameKeywords** (section 2.2.8.7) properties are both not set, the client SHOULD delete the FAI message.

Appropriate actions (3) for the client to perform for each user action are specified in the following table.

User action	Action to perform on each E-mail object	Action to perform on the FAI message object
Ignore	If the E-mail object is not in the Sent Items special folder, move the E-mail object to the Deleted Items special folder.	Set the **PidLidConversationActionMoveFolderEid** property to an array of size 0. Delete the value of the **PidLidConversationActionMoveStoreEid** property (section 2.2.8.4).
Stop Ignoring	If the E-mail object is in the Deleted Items special folder, move the E-mail object to the Inbox folder.	Delete the value of **PidLidConversationActionMoveFolderEid** property and of the **PidLidConversationActionMoveStoreEid** property.
Move to Folder	If the E-mail object is not in the Sent Items special folder, move the E-mail object to the user-specified folder.	Set the **PidLidConversationActionMoveFolderEid** property to the EntryID of the user-specified folder. If the user-specified folder is in the same message store as the FAI message, delete the value of the **PidLidConversationActionMoveStoreEid** property. Otherwise, set the **PidLidConversationActionMoveStoreEid** property to the EntryID of the user-specified message store.
Stop Move	None.	Delete the value of the **PidLidConversationActionMoveFolderEid** property and the **PidLidConversationActionMoveStoreEid** property.
Add Category	Append the user-specified category (3) to the **PidNameKeywords** property of each E-mail object.	Append the user-specified category (3) to the **PidNameKeywords** property of the FAI message.

User action	Action to perform on each E-mail object	Action to perform on the FAI message object
Remove Category	Remove the user-specified category (3) from the **PidNameKeywords** property of each E-mail object. If this operation would leave the **PidNameKeywords** property empty, delete the **PidNameKeywords** property instead.	Remove the user-specified category (3) from the **PidNameKeywords** property of the FAI message. If this operation would leave the **PidNameKeywords** property empty, delete the **PidNameKeywords** property instead.
Clear All Categories	Delete the **PidNameKeywords** property from each E-mail object.	Delete the value of the **PidNameKeywords** property from the FAI message.

3.1.4.9 Reading Navigation Shortcuts

Navigation shortcuts, as specified in section 2.2.9, are stored as FAI messages in the Common Views folder, as specified in [MS-OXOSFLD] section 2.2, within a message database. Clients open this folder and process all messages where the value of the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) is "IPM.Microsoft.WunderBar.Link". Clients SHOULD advise for notifications on this folder to process changes to this table by other clients.

3.1.4.10 Writing Navigation Shortcuts

To write settings in a navigation shortcut, as specified in section 2.2.9, the client MUST open the Common Views folder, as specified in [MS-OXOSFLD] section 2.2. The client MUST save the navigation shortcut message in the Common Views folder.

If the client is creating a new navigation shortcut, the client MUST create a new message by sending the **RopCreateMessage** ROP ([MS-OXCROPS] section 2.2.6.2) on the folder, passing the **Associated** flag. If the client is modifying an existing navigation shortcut, the client MUST open the message for the navigation shortcut by sending the **RopOpenMessage** ROP ([MS-OXCROPS] section 2.2.6.1) with the **ReadWrite** flag in the **OpenModeFlags** field.

The client MUST then send the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) with the relevant properties, as specified in sections 3.1.4.10.1 and 3.1.4.10.2.

The client MUST then send the **RopSaveChangesMessage** ROP ([MS-OXCROPS] section 2.2.6.3) to persist changes to the message.

3.1.4.10.1 Group Headers

Group headers are shortcuts that group other non-group header shortcuts together. Group headers have the following properties:

- **PidTagWlinkGroupHeaderID** (section 2.2.9.3)
- **PidTagWlinkSaveStamp** (section 2.2.9.4)
- **PidTagWlinkType** (section 2.2.9.5)
- **PidTagWlinkFlags** (section 2.2.9.6)

- **PidTagWlinkOrdinal** (section 2.2.9.7)
- **PidTagWlinkFolderType** (section 2.2.9.8)
- **PidTagWlinkSection** (section 2.2.9.14)

3.1.4.10.2 Shortcuts

- Shortcuts have the following properties:
- **PidTagWlinkSaveStamp** (section 2.2.9.4)
- **PidTagWlinkType** (section 2.2.9.5)
- **PidTagWlinkFlags** (section 2.2.9.6)
- **PidTagWlinkOrdinal** (section 2.2.9.7)
- **PidTagWlinkEntryId** (section 2.2.9.8)
- **PidTagWlinkRecordKey** (section 2.2.9.9)
- **PidTagWlinkStoreEntryId** (section 2.2.9.10)
- **PidTagWlinkFolderType** (section 2.2.9.8)
- **PidTagWlinkGroupClsid** (section 2.2.9.12)
- **PidTagWlinkGroupName** (section 2.2.9.13)
- **PidTagWlinkSection** (section 2.2.9.14)
- Shortcuts with the value of the **PidTagWlinkSection** property set to **wbsidCalendar** SHOULD<38> have the following properties if needed or appropriate:
- **PidTagWlinkCalendarColor** (section 2.2.9.15)
- **PidTagWlinkAddressBookEID** (section 2.2.9.16)
- **PidTagWlinkAddressBookStoreEID** (section 2.2.9.17)
- **PidTagWlinkROGroupType** (section 2.2.9.19)
- Shortcuts have the following property available to limit the visibility of a shortcut to a single machine:

  - **PidTagWlinkClientID** (section 2.2.9.18)

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Processing a Conversation Action on Incoming E-mail Objects

When a new E-mail object arrives in a message store, the client SHOULD<39> perform the following steps:

1. The client SHOULD locate the FAI message for the conversation action, as specified in section 2.2.8, in the Conversation Actions Settings special folder with bytes 6-21 of the **PidTagConversationIndex** property (section 2.2.8.8), corresponding to the

PidTagConversationId property ([MS-OXOMSG] section 2.2.1.2) of the incoming E-mail object. If no FAI message is found, the client SHOULD NOT process any conversation actions on the incoming E-mail object.

2. If the **PidLidConversationProcessed** property (section 2.2.8.6) is set on the incoming E-mail object, the client SHOULD execute duplicate detection (as specified in section 3.1.5.1.1) and SHOULD NOT process any conversation actions on the incoming E-mail object.

3. The client SHOULD perform the appropriate actions (3) on the incoming E-mail object based on the properties of the FAI message, as specified in the table that follows these steps. Note that more than one property can apply to a given FAI message.

4. If the server returned a version less than 14.0.324.0 and if step 3 did not encounter an error opening the message store, the **PidLidConversationProcessed** property SHOULD be set to a random number on the incoming E-mail object.

5. The client SHOULD set the current time (in UTC) as the value of the **PidLidConversationActionLastAppliedTime** property (section 2.2.8.1) on the FAI message.

The conversation actions to be performed on the incoming E-mail object based on the properties of the FAI message are specified in the following table.

Properties on the FAI message	Action to perform on the incoming E-mail object
The **PidNameKeywords** property (section 2.2.8.7) is set.	If the server returned a version less than 14.0.324.0, as described in section 1.7, the categories (3) in the FAI message SHOULD be appended to the **PidNameKeywords** property of the incoming E-mail object.
The **PidLidConversationActionMoveStoreEid** property (section 2.2.8.4) is not set and the value of the **PidLidConversationActionMoveFolderEid** property (section 2.2.8.3) is not an array of size zero.	If the server returned a version less than 14.0.324.0, the client SHOULD move the incoming E-mail object to the folder with an EntryID equal to the **PidLidConversationActionMoveFolderEid** property on the FAI message.
The **PidLidConversationActionMoveStoreEid** property is not set and the value of the **PidLidConversationActionMoveFolderEid** property is an array of size zero.	If the server returned a version less than 14.0.324.0, the client SHOULD move the incoming E-mail object to the Deleted Items special folder.
The **PidLidConversationActionMoveStoreEid** property is set and the value of the **PidLidConversationActionMoveFolderEid** property is not an array of size zero.	(Note that this case is not conditioned on the server version) The client SHOULD move the incoming E-mail object to the folder and message store identified by the **PidLidConversationActionMoveFolderEid** and **PidLidConversationActionMoveStoreEid** properties, respectively.
The **PidLidConversationActionMoveStoreEid** property is set and the value of the **PidLidConversationActionMoveFolderEid** property is an array of size zero.	The client SHOULD NOT move the incoming E-mail object.

3.1.5.1.1 Duplicate Detection for Conversation Action Processing

E-mail objects can be duplicated if two clients process a Move to Folder conversation action, as specified in section 2.2.8, on an incoming E-mail object simultaneously. If the server returned a

version less than 14.0.324.0, as described in section 1.7, the client SHOULD<40> perform duplicate detection on incoming E-mail objects.

The client SHOULD<41> compare the **PidTagSearchKey** property ([MS-OXCPRPT] section 2.2.1.9) on the incoming E-mail object to the **PidTagSearchKey** property of all recently delivered E-mail objects in the same folder.

If a matching E-mail object was found, the client SHOULD compare the unsigned values of the **PidLidConversationProcessed** property (section 2.2.8.6) on both E-mail objects and delete the E-mail object with the smaller unsigned value, doing nothing if the unsigned values are equal.

3.1.5.2 Processing a Conversation Action on Outgoing E-mail Objects

When the client saves a copy of an outgoing E-mail object, the client SHOULD<42> perform additional processing on the copy of the E-mail object in the Sent Items special folder.

As specified in section 2.2.8.8, the client SHOULD locate the conversation action FAI message, as specified in section 2.2.8, in the conversation actions Settings special folder with bytes 6-21 of the **PidTagConversationIndex** property (section 2.2.8.8) corresponding to the **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2) of the sent E-mail object. If no FAI message is found, the client SHOULD NOT process any conversation actions on the sent E-mail object.

If the **PidNameKeywords** property (section 2.2.8.7) is set on the FAI message, the categories (3) in the FAI message SHOULD be appended to the **PidNameKeywords** property of the sent E-mail object.

3.1.6 Timer Events

3.1.6.1 Expiration of Conversation Actions

If the server returned a version less than 14.0.324.0, as described in section 1.7, and the elapsed time between the value of the **PidLidConversationActionLastAppliedTime** property (section 2.2.8.1) of the FAI message and the current time is greater than a client-specific<43> duration, the client SHOULD<44> delete the FAI message.

3.1.7 Other Local Events

None.

3.2 Server Details

Clients operate on folders and messages using the protocols specified in [MS-OXCFOLD] and [MS-OXCMSG]. How a server operates on folders and messages is implementation-dependent, but the results of any such operations MUST be exposed to clients in a manner that is consistent with the Configuration Information Protocol.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Reading Configuration Data

If multiple configuration data messages, as specified in section 2.2.2, of the same type are found, the configuration data messages are deemed in conflict and MUST be resolved. If no specific conflict resolution algorithm is available, the server SHOULD pick the message with the earliest creation time stored in the **PidTagCreationTime** property ([MS-OXCMSG] section 2.2.2.3) when opening the configuration data message, and the rest of the configuration data messages SHOULD be deleted from the message store.

3.2.4.1.1 Reading Working Hours

The server is responsible for enforcing permissions that the user grants to the Calendar special folder. If the client tries to access the configuration data message, as specified in section 2.2.2, without the necessary permissions, the server MUST deny access to the message.

3.2.4.1.2 Reading Category List

The server SHOULD<45> limit the size of the XML document that it will parse to 512 kilobytes.

The server is responsible for enforcing permissions that the user grants to the Calendar special folder. If the client tries to access the configuration data message, as specified in section 2.2.2, without the necessary permissions, the server MUST deny access to the message.

3.2.4.2 Writing Configuration Data

If multiple configuration data messages, as specified in section 2.2.2, of the same type are found, the configuration data messages are deemed in conflict and MUST be resolved. If no specific conflict resolution algorithm is available, the server SHOULD pick the message with the earliest modification time stored in the **PidTagLastModificationTime** property ([MS-OXCMSG] section 2.2.2.2) when saving the configuration data message, and the rest of the configuration data messages SHOULD be deleted from the message store.

3.2.4.3 Reading View Definitions

To read the list of available view definitions for a folder as specified in section 2.2.6, the server MUST enumerate all of the view definition FAI messages in the folders, searching for a match on the **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) and the **PidTagViewDescriptorVersion** property ([MS-OXPROPS] section 2.1046) as specified in section 2.2.6.

After the server has built the list of view definition messages, it can select one of them by using the **PidTagViewDescriptorName** property ([MS-OXPROPS] section 2.1044). After it has selected a view definition message, the server MUST read the settings from the **PidTagViewDescriptorBinary** property (section 2.2.6.1) and the **PidTagViewDescriptorStrings** property (section 2.2.6.2) on the message.

3.2.4.4 Reading Folder Flags

Reading and writing each of the subproperties in the folder flags, as specified in section 2.2.7, are triggered by different events.

3.2.4.4.1 Reading ExtendedFolderFlags

The server MUST read the bit flags in the **ExtendedFolderFlags** subproperty before it can display the folder in the UI.

3.2.4.4.2 Reading SearchFolderID

The server MUST read the value of the **SearchFolderID** subproperty from every search folder (2), as specified in [MS-OXOSRCH] section 2.2.4, in the Finder special folder. Any search folder (2) that has this subproperty is a persistent search folder (2), and the server SHOULD display the search folder (2) as such in the UI. Searches with the Finder special folder are further specified in [MS-OXOSRCH] section 3.1.4.1.2. For more details about the Finder special folder, see [MS-OXOSFLD].

3.2.4.5 Writing Folder Flags

In each case where the server needs to write a new value of one of the subproperties to the folder, it MUST preserve the values of any other unmodified subproperties on the folder, as specified in section 2.2.7.

Folder flags are defined in section 2.2.7.

3.2.4.5.1 Writing ExtendedFolderFlags

Any time the user changes one of the display options for a folder, the server MUST rewrite the **ExtendedFolderFlags** subproperty to that folder.

3.2.4.5.2 Writing ToDoFolderVersion

When the server re-creates or resets the criteria on the To-Do Search folder, it MUST set the **ToDoFolderVersion** subproperty on the folder. For more details about the To-Do Search folder, see [MS-OXOSFLD] section 3.1.1.2 and [MS-OXOSFLD] section 3.1.4.1.2.

3.2.4.6 Applying a Category to a Message

The server MUST store any new categories (3) assigned to the **PidNameKeywords** property (section 2.2.8.7).

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Processing a Change to a Conversation Action FAI Message

As specified in section 2.2.8.8, the client locates all E-mail objects in the message store that have a **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2) matching bytes 6-21 of the **PidTagConversationIndex** property (section 2.2.8.8) on the FAI message and that have a **PidTagMessageDeliveryTime** property ([MS-OXOMSG] section 2.2.3.9) value greater than the value of the **PidLidConversationActionMaxDeliveryTime** property (section 2.2.8.2) on the FAI message.

When the client uploads a change to a conversation action FAI message, as specified in section 2.2.8, the server SHOULD<46> perform the appropriate action (3) on all located E-mail objects,

based on the changes to the FAI message, as specified in the following table (more than one action (3) can apply to a given change).

FAI message change	Action to apply on the E-mail object
The value of the **PidLidConversationActionMoveFolderEid** property (section 2.2.8.3) is changed to an array of size 0.	If the E-mail object is not in the Sent Items special folder, move the E-mail object to the Deleted Items special folder.
The value of the **PidLidConversationActionMoveFolderEid** property was an array of size 0, now unset.	If the E-mail object is in the Deleted Items special folder, move the E-mail object to the Inbox folder.
The value of the **PidLidConversationActionMoveFolderEid** property changed to a non-empty array.	If the **PidLidConversationActionMoveStoreEid** property (section 2.2.8.4) is unset and if the E-mail object is not in the Sent Items special folder, move the E-mail object to the folder with an EntryID equal to the new **PidLidConversationActionMoveFolderEid** property on the FAI message. If the **PidLidConversationActionMoveStoreEid** property is set, do nothing.
The value of the **PidLidConversationActionMoveFolderEid** property was a non-empty array, now unset.	None.
The **PidNameKeywords** property (section 2.2.8.7) is unset, set, or modified.	For all categories (3) added in this change, append the category (3) to the **PidNameKeywords** property of each E-mail object. For all categories (3) removed in this change, remove the category (3) from the **PidNameKeywords** property of each E-mail object.

3.2.5.2 Processing a Conversation Action on Incoming E-mail Objects

When a new E-mail object arrives in a message store, the server SHOULD<47> do the following.

1. As specified in section 2.2.8.8, locate the conversation action FAI message, as specified in section 2.2.8, in the conversation actions Settings special folder with bytes 6-21 of **PidTagConversationIndex** property (section 2.2.8.8) corresponding to the **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2) of the incoming E-mail object. If no FAI message is found, the server SHOULD NOT process any conversation actions on the incoming E-mail object.

2. Perform the appropriate actions (3) on the incoming E-mail object based on the properties of the FAI message, as specified in the following table (more than one property can apply to a given FAI message).

Properties on the FAI message	Action to perform on the incoming E-mail object
The **PidNameKeywords** property (section 2.2.8.7) is set.	The categories (3) in the FAI message SHOULD be appended to the **PidNameKeywords** property of the incoming E-mail object.
The **PidLidConversationActionMoveStoreEid** property (section 2.2.8.4) is not set and the value of	The server SHOULD move the incoming E-mail object to the folder with an EntryID equal to the

Properties on the FAI message	Action to perform on the incoming E-mail object
the **PidLidConversationActionMoveFolderEid** property (section 2.2.8.3) is not an array of size zero.	**PidLidConversationActionMoveFolderEid** property on the FAI message.
The **PidLidConversationActionMoveStoreEid** property is not set and the value of the **PidLidConversationActionMoveFolderEid** property is an array of size zero.	The server SHOULD move the incoming E-mail object to the Deleted Items special folder.
The **PidLidConversationActionMoveStoreEid** propertyis set.	The server SHOULD NOT move the incoming E-mail object.

3.2.5.3 Processing a Conversation Action on Outgoing E-mail Objects

When the server saves a copy of an outgoing E-mail object, it SHOULD<48> perform the following additional processing on the copy of the E-mail object in the Sent Items special folder.

1. As specified in section 2.2.8.8, the server SHOULD locate the conversation action FAI message, as specified in section 2.2.8, in the conversation actions Settings special folder with bytes 6-21 of the **PidTagConversationIndex** property (section 2.2.8.8) corresponding to the **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2) of the sent E-mail object. If no FAI message is found, the server SHOULD NOT process any conversation actions on the sent E-mail object.

2. If the **PidNameKeywords** property (section 2.2.8.7) is set on the FAI message, the categories (3) in the FAI message SHOULD be appended to the **PidNameKeywords** property of the sent E-mail object.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Configuration Data

4.1.1 Dictionaries

The following is a sample XML document stored in the **PidTagRoamingDictionary** property (section 2.2.2.2) on a configuration data message, as described in section 2.2.5.1.

```
<?xml version="1.0"?>
<UserConfiguration>
<Info version="Outlook.12"/>
<Data>
<e k="18-piAutoProcess" v="3-True"/>
<e k="18-piRemindDefault" v="9-15"/>
<e k="18-piReminderUpgradeTime" v="9-212864507"/>
<e k="18-OLPrefsVersion" v="9-1"/>
</Data>
</UserConfiguration>
```

4.1.2 Working Hours

The following is a sample XML document stored in the **PidTagRoamingXmlStream** property (section 2.2.2.3) on a configuration data message, as described in section 2.2.5.2.1.

```
<?xml version="1.0"?>
<Root xmlns="WorkingHours.xsd">
<WorkHoursVersion1>
<TimeZone>
<Bias>480</Bias>
<Standard>
<Bias>0</Bias>
<ChangeDate>
<Time>02:00:00</Time>
<Date>0000/11/01</Date>
<DayOfWeek>0</DayOfWeek>
</ChangeDate>
</Standard>
<DaylightSavings>
<Bias>-60</Bias>
<ChangeDate>
<Time>02:00:00</Time>
<Date>0000/03/02</Date>
<DayOfWeek>0</DayOfWeek>
</ChangeDate>
</DaylightSavings>
<Name>Pacific Standard Time</Name>
</TimeZone>
<TimeSlot>
<Start>09:00:00</Start>
<End>17:00:00</End>
</TimeSlot>
<WorkDays>Monday Tuesday Wednesday Thursday Friday</WorkDays>
</WorkHoursVersion1>
</Root>
```

4.1.3 Category List

The following is a sample XML document stored in the **PidTagRoamingXmlStream** property (section 2.2.2.3) on a configuration data message, as described in section 2.2.5.2.2.

```
<?xml version="1.0"?>
<categories default="Red Category"
            lastSavedSession="5"
            lastSavedTime="2007-12-28T03:01:50.429"
            xmlns="CategoryList.xsd">
  <category name="Red Category"
            color="0"
            keyboardShortcut="0"
            usageCount="7"
            lastTimeUsedNotes="1601-01-01T00:00:00.000"
            lastTimeUsedJournal="1601-01-01T00:00:00.000"
            lastTimeUsedContacts="1601-01-01T00:00:00.000"
            lastTimeUsedTasks="1601-01-01T00:00:00.000"
            lastTimeUsedCalendar="2007-11-28T20:05:04.703"
            lastTimeUsedMail="1601-01-01T00:00:00.000"
            lastTimeUsed="2007-11-28T20:05:04.703"
            lastSessionUsed="3"
            guid="{2B7FC69C-7046-44A2-8FF3-007D7467DC82}"/>
  <category name="Blue Category"
            color="7"
            keyboardShortcut="0"
            usageCount="6"
            lastTimeUsedNotes="1601-01-01T00:00:00.000"
            lastTimeUsedJournal="1601-01-01T00:00:00.000"
            lastTimeUsedContacts="1601-01-01T00:00:00.000"
            lastTimeUsedTasks="1601-01-01T00:00:00.000"
            lastTimeUsedCalendar="2007-12-28T03:00:07.102"
            lastTimeUsedMail="1601-01-01T00:00:00.000"
            lastTimeUsed="2007-12-28T03:00:07.102"
            lastSessionUsed="5"
            guid="{33A1EAE3-8E5E-4912-9580-69FC764FEA35}"/>
  <category name="Purple Category"
            color="8"
            keyboardShortcut="0"
            usageCount="7"
            lastTimeUsedNotes="1601-01-01T00:00:00.000"
            lastTimeUsedJournal="1601-01-01T00:00:00.000"
            lastTimeUsedContacts="1601-01-01T00:00:00.000"
            lastTimeUsedTasks="1601-01-01T00:00:00.000"
            lastTimeUsedCalendar="2007-11-28T20:03:06.018"
            lastTimeUsedMail="1601-01-01T00:00:00.000"
            lastTimeUsed="2007-11-28T20:03:06.018"
            lastSessionUsed="3"
            guid="{58AB8B90-BB05-428A-B8D2-F1C93968C144}"/>
  <category name="Green Category"
            color="4"
            keyboardShortcut="0"
            usageCount="7"
            lastTimeUsedNotes="1601-01-01T00:00:00.000"
            lastTimeUsedJournal="1601-01-01T00:00:00.000"
            lastTimeUsedContacts="1601-01-01T00:00:00.000"
            lastTimeUsedTasks="1601-01-01T00:00:00.000"
            lastTimeUsedCalendar="2007-11-28T20:05:19.468"
            lastTimeUsedMail="1601-01-01T00:00:00.000"
```

```
            lastTimeUsed="2007-11-28T20:05:19.468"
            lastSessionUsed="3"
            guid="{B60A1A8C-ECA3-4573-9CD8-842C284DCA59}"/>
  <category name="Orange Category"
            color="1"
            keyboardShortcut="0"
            usageCount="2"
            lastTimeUsedNotes="1601-01-01T00:00:00.000"
            lastTimeUsedJournal="1601-01-01T00:00:00.000"
            lastTimeUsedContacts="1601-01-01T00:00:00.000"
            lastTimeUsedTasks="1601-01-01T00:00:00.000"
            lastTimeUsedCalendar="1601-01-01T00:00:00.000"
            lastTimeUsedMail="1601-01-01T00:00:00.000"
            lastTimeUsed="2007-11-21T00:07:48.517"
            lastSessionUsed="0"
            guid="{F5F57BF3-A188-48D5-A096-863ACACB2D36}"
            renameOnFirstUse="1"/>
  <category name="Yellow Category"
            color="3"
            keyboardShortcut="0"
            usageCount="5"
            lastTimeUsedNotes="1601-01-01T00:00:00.000"
            lastTimeUsedJournal="1601-01-01T00:00:00.000"
            lastTimeUsedContacts="1601-01-01T00:00:00.000"
            lastTimeUsedTasks="1601-01-01T00:00:00.000"
            lastTimeUsedCalendar="2007-11-21T01:04:25.048"
            lastTimeUsedMail="1601-01-01T00:00:00.000"
            lastTimeUsed="2007-11-21T01:04:25.048"
            lastSessionUsed="2"
            guid="{CA791DEF-676C-4177-A839-CAF8878258F0}"/>
  <category name="Black Category"
            color="14"
            keyboardShortcut="0"
            usageCount="6"
            lastTimeUsedNotes="1601-01-01T00:00:00.000"
            lastTimeUsedJournal="1601-01-01T00:00:00.000"
            lastTimeUsedContacts="1601-01-01T00:00:00.000"
            lastTimeUsedTasks="1601-01-01T00:00:00.000"
            lastTimeUsedCalendar="2007-12-14T02:43:30.719"
            lastTimeUsedMail="1601-01-01T00:00:00.000"
            lastTimeUsed="2007-12-14T02:43:30.719"
            lastSessionUsed="4"
            guid="{77EA6484-D31F-496E-AA07-DC4839D4327A}"/>
</categories>
```

4.2 View Definitions

In this example, a client creates a new table view that includes the following 10 columns:

- Importance
- Reminder
- Icon
- Flag Status
- Attachment

- From
- Subject
- Received
- Size
- Categories

When this new view is applied and transported to the server, the **PidTagViewDescriptorBinary** property (section 2.2.6.1) stores the column description data and the **PidTagViewDescriptorStrings** property (section 2.2.6.2) stores the column headers.

4.2.1 PidTagViewDescriptorBinary

The following is the value of the **PidTagViewDescriptorBinary** property (section 2.2.6.1) that represents the view described in section 4.2.

```
0000: 00 00 00 00 00 00 00 00-08 00 00 00 02 00 00 00
0010: 00 00 00 00 0B 00 00 00-08 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00-00 00 00 00 01 00 04 00
0040: 07 00 00 00 00 00 00 00-28 00 00 00 00 00 00 00
0050: 00 00 00 00 00 00 00 00-00 00 00 00 04 00 00 00
0060: 03 00 17 00 12 00 00 00-00 00 00 00 4A 2F 00 00
0070: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0080: 17 00 00 00 0B 00 03 85-12 00 00 00 00 00 00 00
0090: 40 3F 00 00 00 00 00 00-00 00 00 00 34 01 9A 11
00a0: 00 00 00 00 03 85 00 00-08 20 06 00 00 00 00 00
00b0: C0 00 00 00 00 00 00 46-1E 00 1A 00 12 00 00 00
00c0: 00 00 00 00 0A 27 00 00-00 00 00 00 00 00 00 00
00d0: 00 00 00 00 00 00 00 00-1A 00 00 00 03 00 90 10
00e0: 12 00 00 00 00 00 00 00-4A 2F 00 00 00 00 00 00
00f0: 00 00 00 00 00 00 00 00-00 00 00 00 90 10 00 00
0100: 0B 00 1B 0E 12 00 00 00-00 00 00 00 4A 2F 00 00
0110: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0120: 1B 0E 00 00 1E 00 42 00-0C 00 00 00 00 00 00 00
0130: 00 2F 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0140: 00 00 00 00 42 00 00 00-1E 00 37 00 11 00 00 00
0150: 00 00 00 00 00 2F 00 00-00 00 00 00 00 00 00 00
0160: 00 00 00 00 00 00 00 00-37 00 00 00 40 00 06 0E
0170: 10 00 00 00 00 00 00 00-40 2F 00 00 00 00 00 00
0180: 00 00 00 00 00 00 00 00-00 00 00 00 06 0E 00 00
0190: 03 00 08 0E 0C 00 00 00-00 00 00 00 40 27 00 00
01a0: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
01b0: 08 0E 00 00 1E 10 00 00-12 00 00 00 00 00 00 00
01c0: 20 7B 00 00 00 00 00 00-00 00 00 00 34 01 9A 11
01d0: 01 00 00 00 64 A7 22 00-29 03 02 00 00 00 00 00
01e0: C0 00 00 00 00 00 00 46-12 00 00 00 4B 00 65 00
01f0: 79 00 77 00 6F 00 72 00-64 00 73 00 00 00
```

The first 8 bytes are reserved:

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

The next four bytes specify the **Version** field. After the **Version** field is the value for the **ulFlags** field.

```
0008: 08 00 00 00 02 00 00 00
```

Version: 0x00000008

ulFlags: 0x00000002 (VDF_SORTDESCENDING)

The value of this **ulFlags** field means that the view is sorted by descending order.

Next are the **pres** and **cvcd** fields.

```
0010: 00 00 00 00 0B 00 00 00
```

pres: NULL

cvcd: 0x0B

The value 0x0B in the **cvcd** field means 11 columns (including the blank column) are stored in this packet.

Next are the **ivcdSort** and **cCat** fields.

```
0018: 08 00 00 00 00 00 00 00
```

ivcdSort: 0x08

cCat: 0x0

These values for the **ivcdSort** and **cCat** fields mean that the view is sorted by column "Received" and the sort order (3) is descending (as specified by the **ulFlags** field).

cCat is zero; this means that the table is not grouped.

Next is the **ulCatSort** field:

```
0020: 00 00 00 00
```

ulCatSort: 0x0

Because the table is not grouped, the value of the **ulCatSort** field is zero.

The next 24 bytes are reserved.

```
0024-003b
```

reserved

All column information starts from address 003C. Because this view has not defined restriction (1), the buffer does not store any restriction values.

4.2.1.1 Blank Column

The first column is Blank column. The column uses buffer address between 003C and 005F.

```
0030:                                     01 00 04 00
0040: 07 00 00 00 00 00 00 00-28 00 00 00 00 00 00 00
0050: 00 00 00 00 00 00 00 00-00 00 00 00 04 00 00 00
```

PropertyType: 0x0001

```
0030:                                     01 00
```

PropertyID: 0x0004

```
0030:                                           04 00
```

Cx: 0x00000007

```
0040: 07 00 00 00
```

Reserved1:

```
0040:             00 00 00 00
```

Flags: 0x00000028, or (VCDF_BITMAP | VCDF_NOT_SORTABLE)

```
0040:                         28 00 00 00
```

Reserved2:

```
0040:                                     00 00 00 00
0050: 00 00 00 00 00 00 00 00
```

Kind: 0x00000000

```
0050:                         00 00 00 00
```

ID: 0x00000004

```
0050:                                     04 00 00 00
```

4.2.1.2 Column "Importance"

Next in the buffer is the description of the "Importance" column.

```
0060: 03 00 17 00 12 00 00 00-00 00 00 00 4A 2F 00 00
0070: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0080: 17 00 00 00
```

PropertyType: 0x0003

```
0060: 03 00
```

PropertyID: 0x0017 (**PidTagImportance** ([MS-OXCMSG] section 2.2.1.11))

```
0060:       17 00
```

Cx: 0x00000012

```
0060:             12 00 00 00
```

Reserved1: 0x00000000

```
0060:                         00 00 00 00
```

Flags: 0x00002F4A, or (VCDF_BITMAP | VCDF_CENTER_JUSTIFY | VCDF_SORTDLG | VCDF_GROUPDLG | VCDF_SORTDESCENDING | VCDF_RCOLUMNSDLG | VCDF_MOVEABLE | VCDF_COLUMNSDLG)

```
0060:                                     4A 2F 00 00
```

Reserved2: 0x00000000000000000000000000

```
0070: 00 00 00 00 00 00 00 00-00 00 00 00
```

Kind: 0x00000000

```
0070:                                     00 00 00 00
```

ID: 0x00000017

```
0080: 17 00 00 00
```

4.2.1.3 Column "Reminder"

Next in the buffer is the description of the "Reminder" column.

```
0080:             0B 00 03 85-12 00 00 00 00 00 00 00
0090: 40 3F 00 00 00 00 00 00-00 00 00 00 34 01 9A 11
00a0: 00 00 00 00 03 85 00 00-08 20 06 00 00 00 00 00
00b0: C0 00 00 00 00 00 00 46
```

PropertyType: 0x000B

```
0080:             0B 00
```

PropertyID: 0x8503 (**PidLidReminderSet** ([MS-OXORMDR] section 2.2.1.1))

```
0080:                 03 85
```

Cx: 0x00000012

```
0080:                       12 00 00 00
```

Reserved1: 0x00000000

```
0080:                                   00 00 00 00
```

Flags: 0x00003F40, or (VCDF_NAMEDPROP | VCDF_SORTDESCENDING | VCDF-RCOLUMNSDLG | VCDF_SORTDLG | VCDF_GROUPDLG | VCDF_MOVEABLE | VCDF_COLUMNSDLG)

```
0090: 40 3F 00 00
```

Reserved2: 0x119A01340000000000000000

```
0090:             00 00 00 00-00 00 00 00 34 01 9A 11
```

Kind: 0x00000000

```
00a0: 00 00 00 00
```

ID: 0x00008503

```
00a0:             03 85 00 00
```

Guid: {00062008-0000-0000-C000-000000000046}

```
00a0:                         08 20 06 00 00 00 00 00
00b0: C0 00 00 00 00 00 00 46
```

4.2.1.4 Column "Icon"

Next in the buffer is the description of the "Icon" column.

```
00b0:                         1E 00 1A 00 12 00 00 00
00c0: 00 00 00 00 0A 27 00 00-00 00 00 00 00 00 00 00
00d0: 00 00 00 00 00 00 00 00-1A 00 00 00
```

PropertyType: 001E

PropertyID: 0x001A (**PidTagMessageClass** ([MS-OXCMSG] section 2.2.1.3))

Cx: 0x00000012

Flags: 0x0000270A

Kind: 0x00000000

ID: 0x0000001A

4.2.1.5 Column "Flag Status"

Next in the buffer is the description of the "Flag Status" column.

```
00d0:                                     03 00 90 10
00e0: 12 00 00 00 00 00 00 00-4A 2F 00 00 00 00 00 00
00f0: 00 00 00 00 00 00 00 00-00 00 00 00 90 10 00 00
```

PropertyType: 0x0003

PropertyID: 0x1090 (**PidTagFlagStatus** ([MS-OXOFLAG] section 2.2.1.1))

Cx: 0x00000012

Flags: 0x00002F4A

Kind: 0x00000000

ID: 0x00001090

4.2.1.6 Column "Attachment"

Next in the buffer is the description of the "Attachment" column.

```
0100: 0B 00 1B 0E 12 00 00 00-00 00 00 00 4A 2F 00 00
0110: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0120: 1B 0E 00 00
```

PropertyType: 0x000B

PropertyID: 0x0E1B (**PidTagHasAttachments** ([MS-OXCMSG] section 2.2.1.2))

Cx: 0x00000012

Flags: 0x00002F4A

Kind: 0x00000000

ID: 0x00000E1B

4.2.1.7 Column "From"

Next in the buffer is the description of the "From" column.

```
0120:             1E 00 42 00-0C 00 00 00 00 00 00 00
0130: 00 2F 00 00 00 00 00 00-00 00 00 00 00 00 00 00
0140: 00 00 00 00 42 00 00 00
```

PropertyType: 0x001E

PropertyID: 0x0042 (**PidTagSentRepresentingName** ([MS-OXOMSG] section 2.2.1.57))

Cx: 0x0000000C

Flags: 0x00002F00

Kind: 0x00000000

ID: 0x00000042

4.2.1.8 Column "Subject"

Next in the buffer is the description of the "Subject" column.

```
0140:                   1E 00 37 00 11 00 00 00
0150: 00 00 00 00 00 2F 00 00-00 00 00 00 00 00 00 00
0160: 00 00 00 00 00 00 00 00-37 00 00 00
```

PropertyType: 0x001E

PropertyID: 0x0037 (**PidTagSubject** (section 2.2.8.10))

Cx: 0x00000011

Flags: 0x00002F00

Kind: 0x00000000

ID: 0x00000037

4.2.1.9 Column "Received"

Next in the buffer is the description of column "Received".

```
0160:                                     40 00 06 0E
0170: 10 00 00 00 00 00 00 00-40 2F 00 00 00 00 00 00
0180: 00 00 00 00 00 00 00 00-00 00 00 00 06 0E 00 00
```

PropertyType: 0x0040

PropertyID: 0x0E06 (**PidTagMessageDeliveryTime** ([MS-OXOMSG] section 2.2.3.9)

Cx: 0x00000010

Flags: 0x00002F40

Kind: 0x00000000

ID: 0x00000E06

4.2.1.10 Column "Size"

Next in the buffer is the description of the "Size" column.

```
0190: 03 00 08 0E 0C 00 00 00-00 00 00 00 40 27 00 00
01a0: 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
01b0: 08 0E 00 00
```

PropertyType: 0x0003

PropertyID: 0x0E08 (**PidTagMessageSize** ([MS-OXCFOLD] section 2.2.2.2.1.8))

Cx: 0x0000000C

Flags: 0x00002740

Kind: 0x00000000

ID: 0x00000E08

4.2.1.11 Column "Categories"

Next in the buffer is the description of the "categories" column. This is a column that has the named property **PidNameKeywords** (section 2.2.8.7).

```
01b0:             1E 10 00 00-12 00 00 00 00 00 00 00
01c0: 20 7B 00 00 00 00 00 00-00 00 00 00 34 01 9A 11
01d0: 01 00 00 00 64 A7 22 00-29 03 02 00 00 00 00 00
01e0: C0 00 00 00 00 00 00 46-12 00 00 00 4B 00 65 00
01f0: 79 00 77 00 6F 00 72 00-64 00 73 00 00 00
```

PropertyType: 0x101E

PropertyID: 0x0000

Cx: 0x00000012

Flags: 0x00007B20

Kind: 0x00000001

Guid: {00020329-0000-0000-C000-000000000046}

BufferLength: 0x00000012

Buffer: "Keywords"

4.2.2 PidTagViewDescriptorStrings

In this example, the **PidTagViewDescriptorStrings** property (section 2.2.6.2) contains all the column headers delimited by "\n".

```
\nImportance\nReminder\nIcon\nFlag
Status\nAttachment\nFrom\nSubject\nReceived\nSize\nCategories\n
```

Note The value of the **PidTagViewDescriptorStrings** property begins with "\n" because the first string value in this example is an empty string.

4.3 Conversation Actions

4.3.1 A Categorized and Moved Conversation Action

In this example, a user has a conversation entitled "Solidifying our proposal to Fabrikam, Inc." in which the last E-mail object was delivered at 3:31 PM on 2/17/2009, Pacific Standard Time (11:31 PM on 2/17/2009, UTC). The E-mail objects in this conversation share the same value of the **PidTagConversationId** property ([MS-OXOMSG] section 2.2.1.2).

```
0000: B7 A2 B5 C4 AA 65 1C F2-D3 8C 62 8C 0E AF 56 C4
```

Around 3:50 PM, the user adds the categories (3) "Fabrikam" and "Business Proposals" to the conversation. At 3:51 PM, the user moves the conversation to a folder, "FY09 Archive", that has the following EntryID.

```
0000: 00 00 00 00 0C 99 F4 ED-A2 F1 E4 41 B1 5B 9B 25
0010: 10 91 3E 9D 02 81 00 00
```

The folder is in a message store, "Archived Mail", that has the following EntryID.

```
0000: 00 00 00 00 38 A1 BB 10-05 E5 10 1A A1 BB 08 00
0010: 2B 2A 56 C2 00 00 6D 73-70 73 74 2E 64 6C 6C 00
0020: 00 00 00 00 4E 49 54 41-F9 BF B8 01 00 AA 00 37
0030: D9 6E 00 00 00 00 43 00-3A 00 5C 00 44 00 6F 00
```

```
0040: 63 00 75 00 6D 00 65 00-6E 00 74 00 73 00 20 00
0050: 61 00 6E 00 64 00 20 00-53 00 65 00 74 00 74 00
0060: 69 00 6E 00 67 00 73 00-5C 00 61 00 6A 00 61 00
0070: 6D 00 65 00 73 00 5C 00-4C 00 6F 00 63 00 61 00
0080: 6C 00 20 00 53 00 65 00-74 00 74 00 69 00 6E 00
0090: 67 00 73 00 5C 00 41 00-70 00 70 00 6C 00 69 00
00a0: 63 00 61 00 74 00 69 00-6F 00 6E 00 20 00 44 00
00b0: 61 00 74 00 61 00 5C 00-4D 00 69 00 63 00 72 00
00c0: 6F 00 73 00 6F 00 66 00-74 00 5C 00 4F 00 75 00
00d0: 74 00 6C 00 6F 00 6F 00-6B 00 5C 00 41 00 72 00
00e0: 63 00 68 00 69 00 76 00-65 00 64 00 20 00 4D 00
00f0: 61 00 69 00 6C 00 2E 00-70 00 73 00 74 00 00 00
```

The resulting FAI message in the conversation action Settings special folder will have the following properties.

Property name	Value
PidLidConversationActionLastAppliedTime (section 2.2.8.1)	23:51:11 2/17/2009
PidLidConversationActionMaxDeliveryTime (section 2.2.8.2)	23:31:42 2/17/2009
PidLidConversationActionMoveFolderEid (section 2.2.8.3)	0000: 00 00 00 00 0C 99 F4 ED-A2 F1 E4 41 B1 5B 9B 25 0010: 10 91 3E 9D 02 81 00 00
PidLidConversationActionMoveStoreEid (section 2.2.8.4)	0000: 00 00 00 00 38 A1 BB 10-05 E5 10 1A A1 BB 08 00 0010: 2B 2A 56 C2 00 00 6D 73-70 73 74 2E 64 6C 6C 00 0020: 00 00 00 00 4E 49 54 41-F9 BF B8 01 00 AA 00 37 0030: D9 6E 00 00 00 00 43 00-3A 00 5C 00 44 00 6F 00 0040: 63 00 75 00 6D 00 65 00-6E 00 74 00 73 00 20 00 0050: 61 00 6E 00 64 00 20 00-53 00 65 00 74 00 74 00 0060: 69 00 6E 00 67 00 73 00-5C 00 61 00 6A 00 61 00 0070: 6D 00 65 00 73 00 5C 00-4C 00 6F 00 63 00 61 00 0080: 6C 00 20 00 53 00 65 00-74 00 74 00 69 00 6E 00 0090: 67 00 73 00 5C 00 41 00-70 00 70 00 6C 00 69 00 00a0: 63 00 61 00 74 00 69 00-6F 00 6E 00 20 00 44 00 00b0: 61 00 74 00 61 00 5C 00-4D 00 69 00 63 00 72 00 00c0: 6F 00 73 00 6F 00 66 00-74 00 5C 00 4F 00 75 00 00d0: 74 00 6C 00 6F 00 6F 00-6B 00 5C 00

Property name	Value
	41 00 72 00 00e0: 63 00 68 00 69 00 76 00-65 00 64 00 20 00 4D 00 00f0: 61 00 69 00 6C 00 2E 00-70 00 73 00 74 00 00 00
PidLidConversationActionVersion (section 2.2.8.5)	0x003CCCCC
PidNameKeywords (section 2.2.8.7)	{"Fabrikam", "Business Proposals"}
PidTagConversationIndex (section 2.2.8.8)	0000: 01 00 00 00 00 00 B7 A2- B5 C4 AA 65 1C F2 D3 8C 0000: 62 8C 0E AF 56 C4
PidTagMessageClass (section 2.2.8.9)	"IPM.ConversationAction"
PidTagSubject (section 2.2.8.10)	"Conv.Action: Solidifying our proposal to Fabrikam, Inc."

4.4 Navigation Shortcut

4.4.1 Group Header

Ryan Gregg creates a new group header named "My Work Calendars" to group his shortcuts in his e-mail client application.

The client sends a **RopCreateMessage** ROP request ([MS-OXCROPS] section 2.2.6.2) that has a value for the **FolderId** field set to the ID of the Common Views folder and the **Associated** flag set to 1 and then waits for the server to respond. The server response contains a handle to the **Message object**.

The client uses the Session ID 0x12345678 (generated at random on initialization) and generates a GUID for this shortcut: 5BA943D8DAAA462CA63E9136F65C8681.

The client then sends a **RopSetProperties** ROP request ([MS-OXCROPS] section 2.2.8.6) to set the following properties. The property data types are all defined in ([MS-OXCDATA] section 2.11.1).

Property name	PropertyID	Property type	Value
PidTagMessageClass ([MS-OXCMSG] section 2.2.1.3)	0x001A	0x001F (**PtypString**)	"IPM.Microsoft.WunderBar.Link"
PidTagNormalizedSubject (section 2.2.9.2)	0x0E1D	0x001F (**PtypString**)	"My Work Calendars"
PidTagWlinkGroupHeaderID (section 2.2.9.3)	0x6842	0x0048 (**PtypGuid**)	5BA943D8DAAA462C A63E9136F65C8681
PidTagWlinkSaveStamp (section 2.2.9.4)	0x6847	0x0003 (**PtypInteger32**)	0x12345678
PidTagWlinkType (section 2.2.9.5)	0x6849	0x0003 (**PtypInteger32**)	0x00000004
PidTagWlinkFlags (section	0x684A	0x0003	0x00000000

Property name	PropertyID	Property type	Value
2.2.9.6)		(**PtypInteger32**)	
PidTagWlinkOrdinal (section 2.2.9.7)	0x684B	0x0102 (**PtypBinary**)	0x80
PidTagWlinkFolderType (section 2.2.9.11)	0x684F	0x0048 (**PtypGuid**)	0278060000000000 C000000000000046
PidTagWlinkSection (section 2.2.9.14)	0x6852	0x0003 (**PtypInteger32**)	0x00000003

The client then sends a **RopSaveChangesMessage** ROP request ([MS-OXCROPS] section 2.2.6.3) to persist the object on the server, and a **RopRelease** ROP request ([MS-OXCROPS] section 2.2.15.3) to release the object.

4.4.2 Navigation Shortcut

Ryan creates a new shortcut to his folder "Meetings" and wants to group it under the new "My Work Calendars" group he created in the example in section 4.4.1.

The client sends a **RopCreateMessage ROP request** ([MS-OXCROPS] section 2.2.6.2) that has the FolderId set to the ID of the Common Views folder and the **Associated** flag set to 1 and then waits for the server to respond. The server response contains a handle to the Message object.

The client then sends a **RopSetProperties** ROP request ([MS-OXCROPS] section 2.2.8.6) to set the following properties. The property data types are all defined in ([MS-OXCDATA] section 2.11.1).

Property name	PropertyID	Property type	Value
PidTagMessageClass ([MS-OXCMSG] section 2.2.1.3)	0x001A	0x001F (**PtypString**)	"IPM.Microsoft.WunderBar.Link"
PidTagNormalizedSubject (section 2.2.9.2)	0x0E1D	0x001F (**PtypString**)	"Meetings"
PidTagWlinkSaveStamp (section 2.2.9.4)	0x6847	0x0003 (**PtypInteger32**)	0x12345678
PidTagWlinkType (section 2.2.9.5)	0x6849	0x0003 (**PtypInteger32**)	0x00000000
PidTagWlinkFlags (section 2.2.9.6)	0x684A	0x0003 (**PtypInteger32**)	0x00000000
PidTagWlinkOrdinal (section 2.2.9.7)	0x684B	0x0102 (**PtypBinary**)	0x80
PidTagWlinkEntryId (section 2.2.9.8)	0x684C	0x0102 (**PtypBinary**)	See explanation following this table
PidTagWlinkRecordKey (section 2.2.9.9)	0x684D	0x0102 (**PtypBinary**)	See explanation following this table
PidTagWlinkStoreEntryId (section 2.2.9.10)	0x684E	0x0102 (**PtypBinary**)	See explanation following this table

Property name	PropertyID	Property type	Value
PidTagWlinkFolderType (section 2.2.9.11)	0x684F	0x0048 (**PtypGuid**)	0278060000000000 C000000000000046
PidTagWlinkGroupClsid (section 2.2.9.12)	0x6850	0x0048 (**PtypGuid**)	5BA943D8DAAA462C A63E9136F65C8681
PidTagWlinkGroupName (section 2.2.9.13)	0x6851	0x001F (**PtypString**)	"My Work Calendars"
PidTagWlinkSection (section 2.2.9.14)	0x6852	0x0003 (**PtypInteger32**)	0x00000003

The values of the **PidTagWlinkEntryId**, **PidTagWlinkRecordKey**, and **PidTagWlinkStoreEntryId** properties are copied directly from the corresponding properties of the actual Calendar folder as described in section 2.2.9.8, section 2.2.9.9, and section 2.2.9.10.

The client then sends a **RopSaveChangesMessage** ROP request ([MS-OXCROPS] section 2.2.6.3) to persist the object on the server and a **RopRelease** ROP request ([MS-OXCROPS] section 2.2.15.3) to release the object.

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the Configuration Information protocol. General security considerations pertaining to the underlying transport apply and are described in [MS-OXCMSG].

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

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

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

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

<1> Section 1.7: Exchange 2003, Exchange 2007, Exchange 2010, the initial release version of Exchange 2013, Office Outlook 2003, Office Outlook 2007, Outlook 2010, and the initial release version of Outlook 2013 do not support the **X-ServerApplication** header and the **Connect** request type. The **X-ServerApplication** header and the **Connect** request type were introduced in Exchange 2013 SP1 and Outlook 2013 SP1.

<2> Section 2.2.3: In Office Outlook 2007, Outlook 2010, and Outlook 2013 Cached Mode, when the local cached copy of the category list is in conflict with the copy on the server, Office Outlook 2007, Outlook 2010, and Outlook 2013 sometimes do not escape the single quote character.

<3> Section 2.2.5: Exchange 2003 and Office Outlook 2003 do not read or write configuration data FAI messages.

<4> Section 2.2.5.1: Exchange 2007 uses the string "Exchange.12", and Office Outlook 2007 uses the string "Outlook.12" for the **version** attribute. Exchange 2010 uses the string "Exchange.14", and Outlook 2010 uses the string "Outlook.14" for the **version** attribute. Exchange 2013 uses the string "Exchange.15", and Outlook 2013 uses the string "Outlook.15" for the **version** attribute.

<5> Section 2.2.5.1.1: Office Outlook 2003 and Office Outlook 2007 do not support the **piAutoDeleteReceipts** property.

<6> Section 2.2.5.2.1: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 neither read nor write the **Name** element.

<7> Section 2.2.5.2.2: Office Outlook 2007, Outlook 2010, and Outlook 2013 ignore any digits after the first 3, which means that the maximum precision of elements of the **dateTimeRestrictedType** is milliseconds.

<8> Section 2.2.5.2.2: This restriction (1) on the category name is supported only by Exchange 2003.

<9> Section 2.2.5.2.2: Exchange 2003 does not use the **keyboardShortcut** attribute data.

<10> Section 2.2.5.2.2: Office Outlook 2007, Outlook 2010, and Outlook 2013 write the usage count as indicated. However, the most frequently used list is not implemented, and the **usageCount** attribute is not used.

<11> Section 2.2.5.2.2: Office Outlook 2007, Outlook 2010, and Outlook 2013 will write the last session that this category (3) was applied or changed by the user, but this value is not used.

<12> Section 2.2.5.2.2: Exchange 2003 and Office Outlook 2003 SP3 do not support renaming categories (3).

<13> Section 2.2.5.2.2: Office Outlook 2007, Outlook 2010, and Outlook 2013 will increment this number on every session. A new session occurs for these clients either when the user starts Office Outlook 2007, Outlook 2010, or Outlook 2013 or when not less than 12 hours have elapsed since the previous session.

<14> Section 2.2.5.2.3: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not store the settings defined in this section.

<15> Section 2.2.5.2.3: Exchange 2010 and Exchange 2013 always set the value of the **OptedInto** attribute to "False".

<16> Section 2.2.6: View definitions are used only by Office Outlook 2003 and Office Outlook 2007.

<17> Section 2.2.6.1: Outlook 2010 and Outlook 2013 do not read or write the **PidTagViewDescriptorBinary** property ([MS-OXPROPS] section 2.1043).

<18> Section 2.2.6.2: Outlook 2010 and Outlook 2013 do not read or write the **PidTagViewDescriptorStrings** property ([MS-OXPROPS] section 2.1045).

<19> Section 2.2.7.1.2: Exchange 2003 and Office Outlook 2003 do not support displaying the policy description.

<20> Section 2.2.8: Conversation actions are not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<21> Section 2.2.8.1: The **PidLidConversationActionLastAppliedTime** property is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<22> Section 2.2.8.2: The **PidLidConversationActionMaxDeliveryTime** property is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<23> Section 2.2.8.3: The **PidLidConversationActionMoveFolderEid** property is not supported Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<24> Section 2.2.8.4: The **PidLidConversationActionMoveStoreEid** property is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<25> Section 2.2.8.5: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not set the **PidLidConversationActionVersion** property and ignore it if it is set.

<26> Section 2.2.8.7: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not set the **PidNameKeywords** property for conversation actions.

<27> Section 2.2.8.8: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not set the **PidTagConversationIndex** property for conversation actions.

<28> Section 2.2.8.9: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not set the **PidTagMessageClass** property to "IPM.ConversationAction".

<29> Section 2.2.8.10: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not set the **PidTagSubject** property for conversation actions.

<30> Section 2.2.9: Navigation shortcuts are not supported by Exchange 2003, Exchange 2007, or Office Outlook 2003.

<31> Section 2.2.9.16: The **PidTagWlinkAddressBookEID** property is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007. Outlook 2010 and Outlook 2013 set the **PidTagWlinkAddressBookEID** property on calendar shortcuts only.

<32> Section 2.2.9.17: The **PidTagWlinkAddressBookStoreEID** property is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007. Outlook 2010 and Outlook 2013 set the **PidTagWlinkAddressBookStoreEID** property on calendar shortcuts only.

<33> Section 2.2.9.18: The shortcut behavior is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<34> Section 2.2.9.19: The **PidTagWlinkROGroupType** property is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<35> Section 3.1.4.1.2: Office Outlook 2007, Outlook 2010, and Outlook 2013 use the default values of working from 8 A.M. to 5 P.M., Monday – Friday, in the user's current system time zone.

<36> Section 3.1.4.8: Performing conversation actions is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<37> Section 3.1.4.8: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not support finding matching E-mail objects related to a conversation. When connected to an online-mode server and performing an Ignore, Stop Ignoring, Move to Folder, or Stop Move conversation action, Outlook 2010 and Outlook 2013 will not attempt to locate matching E-mail objects. For all other scenarios, Outlook 2010 and Outlook 2013 will attempt to locate matching E-mail objects.

<38> Section 3.1.4.10.2: Setting the properties specified in this section is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007. Whether Outlook 2010 and Outlook 2013 set the **PidTagWlinkCalendarColor** property (section 2.2.9.15), the **PidTagWlinkAddressBookEID** property (section 2.2.9.16), the **PidTagWlinkAddressBookStoreEID** property (section 2.2.9.17), and the **PidTagWlinkROGroupType** property (section 2.2.9.19) depends upon configuration or administrative settings.

<39> Section 3.1.5.1: Performing conversation actions on incoming messages is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<40> Section 3.1.5.1.1: Duplicate detection on incoming E-mail objects is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<41> Section 3.1.5.1.1: Comparing the value of the **PidTagSearchKey** property ([MS-OXPROPS] section 2.986) on incoming E-mail objects is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007. Outlook 2010 and Outlook 2013 maintain a cache of all E-mail objects received in the past 15 minutes.

<42> Section 3.1.5.2: Performing additional processing on the copy of the E-mail object in the Sent Items special folder is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<43> Section 3.1.6.1: Office Outlook 2003, Office Outlook 2007, Outlook 2010, and Outlook 2013 have the default duration of 336 hours (14 days).

<44> Section 3.1.6.1: Office Outlook 2003, Office Outlook 2007, Outlook 2010, and Outlook 2013 implement conversation action expiration without timers. Instead, every time an algorithm calls for an FAI message to be located, Office Outlook 2003, Office Outlook 2007, Outlook 2010, and Outlook 2013 run the expiration logic before proceeding with the original algorithm.

<45> Section 3.2.4.1.2: Exchange 2007, Exchange 2010, and Exchange 2013 will stop reading the XML document beyond 512 kilobytes.

<46> Section 3.2.5.1: Processing in response to a client uploading changes to a conversation action FAI message is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

<47> Section 3.2.5.2: Exchange 2003, Exchange 2007, Office Outlook 2003, and Office Outlook 2007 do not support processing conversation actions on incoming E-mail objects.

<48> Section 3.2.5.3: The additional processing on the copy of the E-mail object in the Sent Items special folder specified in this section is not supported by Exchange 2003, Exchange 2007, Office Outlook 2003, or Office Outlook 2007.

7 Change Tracking

This section identifies changes that were made to the [MS-OXOCFG] protocol document between the July 2013 and November 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

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

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

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

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

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

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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

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

- Obsolete document removed.

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

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

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

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2.2 Informative References	Added [MS-OXCMAPIHTTP] to the list of informative references.	Y	Content updated.
1.7 Versioning and Capability Negotiation	Added product behavior note specifying support for the X-ServerApplication header and the Connect request type.	Y	New product behavior note added.
1.7 Versioning and Capability Negotiation	Revised to also specify that version number is returned in the X-ServerApplication header of Connect request type response, as described in [MS-OXCMAPIHTTP].	Y	Content updated.
3.2 Server Details	Revised to reference [MS-OXCFOLD] and [MS-OXCMSG] instead of [MS-OXCRPC] for details about operations on folders and messages.	N	Content updated.

8 Index

A

Abstract data model
  client 65
  server 77
Applicability 13

B

Binary Format message 16

C

Capability negotiation 13
Change tracking 103
Client
  abstract data model 65
  initialization - navigation shortcuts 65
  other local events 77
  timers 65
Client - higher layer triggered events
  applying a category to a message 72
  performing a conversation action 72
  reading configuration data 65
  reading folder flags 71
  reading navigation shortcuts 74
  reading view definitions 69
  writing configuration data 68
  writing folder flags 71
  writing view definitions 69
Client - message processing
  processing a conversation action on incoming E-mail objects 75
  processing a conversation action on outgoing E-mail objects 77
Client - sequencing rules
  processing a conversation action on incoming E-mail objects 75
  processing a conversation action on outgoing E-mail objects 77
Client - timer events - expiration of conversation actions 77
Configuration data
  dictionaries 16
  XML streams 20
Configuration data example
  category list 83
  dictionaries 82
  working hours 82
Configuration Data message 16
Configuration data properties
  PidTagRoamingDatatypes property 14
  PidTagRoamingDictionary property 15
  PidTagRoamingXmlStream property 15
Configuration Data Properties message 14
Conversation actions example - a categorized and moved conversation action 93
Conversation Actions message 56
  PidLidConversationActionLastAppliedTime property 56
  PidLidConversationActionMaxDeliveryTime property 56
  PidLidConversationActionMoveFolderEid property 57
  PidLidConversationActionMoveStoreEid property 57
  PidLidConversationActionVersion property 57
  PidLidConversationProcessed property 58
  PidNameKeywords property 58
  PidTagConversationIndex property 58
  PidTagMessageClass property 58
  PidTagSubject property 59

D

Data model - abstract
  client 65
  server 77
Dictionaries configuration data 16

E

Examples - configuration data
  category list 83
  dictionaries 82
  working hours 82
Examples – conversation actions - a categorized and moved conversation action 93
Examples – navigation shortcut
  group header 95
  navigation shortcut 96
Examples – view definitions
  overview 84
  PidTagViewDescriptorBinary 85
  PidTagViewDescriptorStrings 93

F

Fields - vendor-extensible 13
Folder Flags message 53
Folder flags message - sub-property 53

G

Glossary 8

H

Higher layer triggered events - client
  applying a category to a message 72
  performing a conversation action 72
  reading configuration data 65
  reading folder flags 71
  reading navigation shortcuts 74
  reading view definitions 69
  writing configuration data 68

writing folder flags 71
writing view definitions 69
Higher layer triggered events - server
applying a category to a message 79
reading configuration data 78
reading folder flags 79
reading view definitions 78
writing configuration data 78
writing folder flags 79

I

Implementer - security considerations 98
Index of security parameters 98
Informative references 10
Initialization
client - navigation shortcuts 65
server 78
Introduction 8

M

Message processing - client
processing a conversation action on incoming E-mail objects 75
processing a conversation action on outgoing E-mail objects 77
Message processing - server
processing a change to a conversation action FAI message 79
processing a conversation action on incoming E-mail objects 80
processing a conversation action on outgoing E-mail objects 81
Message syntax 14
Messages
Binary Format 16
Configuration Data 16
Configuration Data Properties 14
Conversation Actions 56
Folder Flags 53
message syntax 14
Namespaces 14
Navigation Shortcuts 59
transport 14
View Definitions 34
XML Format 15

N

Namespaces message 14
Navigation shortcut example
group header 95
navigation shortcut 96
Navigation Shortcuts message 59
PidTagMessageClass property 59
PidTagNormalizedSubject property 59
PidTagWlinkAddressBookEID property 63
PidTagWlinkAddressBookStoreEID property 63
PidTagWlinkCalendarColor property 63
PidTagWlinkClientID property 64
PidTagWlinkEntryId property 61
PidTagWlinkFlags property 60
PidTagWlinkFolderType property 61
PidTagWlinkGroupClsid property 62
PidTagWlinkGroupHeaderID property 59
PidTagWlinkGroupName property 62
PidTagWlinkOrdinal property 61
PidTagWlinkRecordKey property 61
PidTagWlinkROGroupType property 64
PidTagWlinkSaveStamp property 59
PidTagWlinkSection property 62
PidTagWlinkStoreEntryId property 61
PidTagWlinkType property 60
Normative references 9

O

Other local events
client 77
server 81
Overview (synopsis) 11

P

Parameters - security index 98
PidLidConversationActionLastAppliedTime property conversation actions message 56
PidLidConversationActionMaxDeliveryTime property conversation actions message 56
PidLidConversationActionMoveFolderEid property conversation actions message 57
PidLidConversationActionMoveStoreEid property conversation actions message 57
PidLidConversationActionVersion property conversation actions message 57
PidLidConversationProcessed property conversation actions message 58
PidNameKeywords property conversation actions message 58
PidTagConversationIndex property conversation actions message 58
PidTagMessageClass property conversation actions message 58
PidTagMessageClass property navigation shortcuts message 59
PidTagNormalizedSubject property navigation shortcuts message 59
PidTagRoamingDatatypes configuration data property 14
PidTagRoamingDictionary configuration data property 15
PidTagRoamingXmlStream configuration data property 15
PidTagSubject property conversation actions message 59
PidTagViewDescriptorBinary property view definitions message 35
PidTagViewDescriptorStrings property view definitions message 52
PidTagWlinkAddressBookEID property navigation shortcuts message 63
PidTagWlinkAddressBookStoreEID property navigation shortcuts message 63

PidTagWlinkCalendarColor property navigation shortcuts message 63
PidTagWlinkClientID property navigation shortcuts message 64
PidTagWlinkEntryId property navigation shortcuts message 61
PidTagWlinkFlags property navigation shortcuts message 60
PidTagWlinkFolderType property navigation shortcuts message 61
PidTagWlinkGroupClsid property navigation shortcuts message 62
PidTagWlinkGroupHeaderID property navigation shortcuts message 59
PidTagWlinkGroupName property navigation shortcuts message 62
PidTagWlinkOrdinal property navigation shortcuts message 61
PidTagWlinkRecordKey property navigation shortcuts message 61
PidTagWlinkROGroupType property navigation shortcuts message 64
PidTagWlinkSaveStamp property navigation shortcuts message 59
PidTagWlinkSection property navigation shortcuts message 62
PidTagWlinkStoreEntryId property navigation shortcuts message 61
PidTagWlinkType property navigation shortcuts message 60
Preconditions 13
Prerequisites 13
Product behavior 99

R

References 9
  informative 10
  normative 9
Relationship to other protocols 11

S

Security
  implementer considerations 98
  parameter index 98
Sequencing rules - client
  processing a conversation action on incoming E-mail objects 75
  processing a conversation action on outgoing E-mail objects 77
Sequencing rules - server
  processing a change to a conversation action FAI message 79
  processing a conversation action on incoming E-mail objects 80
  processing a conversation action on outgoing E-mail objects 81
Server
  abstract data model 77
  initialization 78
  other local events 81
  overview 77
  timer events 81
  timers 77
Server - higher layer triggered events
  applying a category to a message 79
  reading configuration data 78
  reading folder flags 79
  reading view definitions 78
  writing configuration data 78
  writing folder flags 79
Server - message processing
  processing a change to a conversation action FAI message 79
  processing a conversation action on incoming E-mail objects 80
  processing a conversation action on outgoing E-mail objects 81
Server - sequencing rules
  processing a change to a conversation action FAI message 79
  processing a conversation action on incoming E-mail objects 80
  processing a conversation action on outgoing E-mail objects 81
Standards assignments 13
Sub-property folder flags message 53

T

Timer events
  client - expiration of conversation actions 77
  server 81
Timers
  client 65
  server 77
Tracking changes 103
Transport 14
Triggered events - client
  applying a category to a message 72
  performing a conversation action 72
  reading configuration data 65
  reading folder flags 71
  reading navigation shortcuts 74
  reading view definitions 69
  writing configuration data 68
  writing folder flags 71
  writing view definitions 69
Triggered events - server
  applying a category to a message 79
  reading configuration data 78
  reading folder flags 79
  reading view definitions 78
  writing configuration data 78
  writing folder flags 79

V

Vendor-extensible fields 13
Versioning 13
View definitions example
  overview 84
  PidTagViewDescriptorBinary 85

PIdTagViewDescriptorStrings 93
View Definitions message 34
PidTagViewDescriptorBinary property 35
PidTagViewDescriptorStrings property 52

X

XML Format message 15
XML streams configuration data 20