

[MS-OXOCFG]: Configuration Information Protocol Specification

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's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>) or the Community Promise (available here: <http://www.microsoft.com/interop/cp/default.aspx>). 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.
- **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.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability.
Microsoft Corporation	April 25, 2008	0.2	Revised and updated property names and other technical content.
Microsoft Corporation	June 27, 2008	1.0	Initial Release.
Microsoft Corporation	August 6, 2008	1.01	Revised and edited technical content.
Microsoft Corporation	September 3, 2008	1.02	Revised and edited technical content.
Microsoft Corporation	December 3, 2008	1.03	Revised and edited technical content.
Microsoft Corporation	April 10, 2009	2.0	Updated technical content for new product releases.

Table of Contents

1	Introduction	7
1.1	Glossary	7
1.2	References	9
1.2.1	Normative References	9
1.2.2	Informative References	10
1.3	Protocol Overview	10
1.3.1	Configuration Data	11
1.3.2	View Definitions	11
1.3.3	Folder Flags	12
1.3.4	Conversation Actions	12
1.4	Relationship to Other Protocols	12
1.5	Prerequisites/Preconditions	13
1.6	Applicability Statement	14
1.7	Versioning and Capability Negotiation	14
1.8	Vendor-Extensible Fields	14
1.9	Standards Assignments	14
2	Messages	14
2.1	Transport	14
2.2	Message Syntax	14
2.2.1	XML Format	14
2.2.2	Binary Format	15
2.2.3	Configuration Data	15
2.2.3.1	Dictionaries	16
2.2.3.1.1	Calendar Options	19
2.2.3.2	XML Streams	21
2.2.3.2.1	Working Hours	21
2.2.3.2.2	Category List	25
2.2.4	View Definitions	31
2.2.4.1	PidTagViewDescriptorBinary	32
2.2.4.1.1	ColumnPacket	35
2.2.4.1.2	RestrictionPacket	39
2.2.4.1.3	RestrictionBlock	39
2.2.4.2	PidTagViewDescriptorStrings	60
2.2.5	Folder Flags	61
2.2.6	Conversation Actions	64
2.2.6.1	PidLidConversationActionLastAppliedTime	65
2.2.6.2	PidLidConversationActionMaxDeliveryTime	65
2.2.6.3	PidLidConversationActionMoveFolderEid	65
2.2.6.4	PidLidConversationActionMoveStoreEid	65
2.2.6.5	PidLidConversationActionVersion	66
2.2.6.6	PidNameKeywords	66

2.2.6.7	PidTagConversationIndex.....	66
2.2.6.8	PidTagMessageClass.....	67
2.2.6.9	PidTagSubject.....	67
2.2.7	Navigation Shortcuts.....	67
2.2.7.1	PidTagMessageClass.....	67
2.2.7.2	PidTagNormalizedSubject.....	67
2.2.7.3	PidTagWlinkGroupHeaderID.....	67
2.2.7.4	PidTagWlinkSaveStamp.....	68
2.2.7.5	PidTagWlinkType.....	68
2.2.7.6	PidTagWlinkFlags.....	68
2.2.7.7	PidTagWlinkOrdinal.....	69
2.2.7.8	PidTagWlinkEntryId.....	70
2.2.7.9	PidTagWlinkRecordKey.....	70
2.2.7.10	PidTagWlinkStoreEntryId.....	70
2.2.7.11	PidTagWlinkFolderType.....	70
2.2.7.12	PidTagWlinkGroupClsid.....	71
2.2.7.13	PidTagWlinkGroupName.....	71
2.2.7.14	PidTagWlinkSection.....	71
2.2.7.15	PidTagWlinkCalendarColor.....	71
2.2.7.16	PidTagWlinkAddressBookEID.....	72
2.2.7.17	PidTagWlinkAddressBookStoreEID.....	72
2.2.7.18	PidTagWlinkClientID.....	73
2.2.7.19	PidTagWlinkROGroupType.....	73
3	Protocol Details.....	73
3.1	Client Details.....	73
3.1.1	Abstract Data Model.....	73
3.1.2	Timers.....	74
3.1.3	Initialization.....	74
3.1.3.1	Navigation Shortcuts.....	74
3.1.4	Higher-Layer Triggered Events.....	74
3.1.4.1	Reading Configuration Data.....	74
3.1.4.1.1	Reading Dictionaries.....	75
3.1.4.1.2	Reading Working Hours.....	76
3.1.4.1.3	Reading Category List.....	76
3.1.4.2	Writing Configuration Data.....	77
3.1.4.2.1	Writing Dictionaries.....	77
3.1.4.2.2	Writing Working Hours.....	78
3.1.4.2.3	Writing Category List.....	78
3.1.4.3	Writing View Definitions.....	79
3.1.4.4	Reading Folder Flags.....	80
3.1.4.4.1	Reading ExtendedFolderFlags.....	80
3.1.4.4.2	Reading SearchFolderID.....	80
3.1.4.4.3	Reading ToDoFolderVersion.....	80

3.1.4.5	Writing Folder Flags	80
3.1.4.5.1	Writing ExtendedFolderFlags	80
3.1.4.5.2	Writing SearchFolderID	80
3.1.4.5.3	Writing ToDoFolderVersion	81
3.1.4.6	Applying a category to a Message	81
3.1.4.7	Performing a Conversation Action	81
3.1.4.8	Reading Navigation Shortcuts	83
3.1.4.9	Writing Navigation Shortcuts	83
3.1.4.9.1	Group Headers	83
3.1.4.9.2	Shortcuts	84
3.1.5	Message Processing Events and Sequencing Rules	84
3.1.5.1	Processing a Conversation Action on Incoming E-mail Objects	84
3.1.5.1.1	Duplicate Detection for Conversation Action Processing	85
3.1.5.2	Processing a Conversation Action on Outgoing E-mail Objects	86
3.1.6	Timer Events	86
3.1.6.1	Expiration of Conversation Actions	86
3.1.7	Other Local Events	86
3.2	Server Details	86
3.2.1	Abstract Data Model	87
3.2.2	Timers	87
3.2.3	Initialization	87
3.2.4	Higher-Layer Triggered Events	87
3.2.4.1	Reading Configuration Data	87
3.2.4.1.1	Reading Working Hours	87
3.2.4.1.2	Reading Category List	87
3.2.4.2	Writing Configuration Data	87
3.2.4.3	Reading View Definitions	88
3.2.4.4	Reading Folder Flags	88
3.2.4.4.1	Reading ExtendedFolderFlags	88
3.2.4.4.2	Reading SearchFolderID	88
3.2.4.5	Writing Folder Flags	88
3.2.4.5.1	Writing ExtendedFolderFlags	88
3.2.4.5.2	Writing ToDoFolderVersion	88
3.2.4.6	Applying a category to a Message	88
3.2.5	Message Processing Events and Sequencing Rules	89
3.2.5.1	Processing a Change to a Conversation Action FAI Message	89
3.2.5.2	Processing a Conversation Action on Incoming E-mail Objects	90
3.2.5.3	Processing a Conversation Action on Outgoing E-mail Objects	90
3.2.6	Timer Events	91
3.2.7	Other Local Events	91
4	Protocol Examples	91
4.1	Configuration Data	91
4.1.1	Dictionaries	91

4.1.2	Working Hours	91
4.1.3	Category List	92
4.2	View Definitions	94
4.2.1	PidTagViewDescriptorBinary	94
4.2.1.1	Blank Column	96
4.2.1.2	Column "Importance"	97
4.2.1.3	Column "Reminder"	98
4.2.1.4	Column "Icon"	99
4.2.1.5	Column "Flag Status"	99
4.2.1.6	Column "Attachment"	100
4.2.1.7	Column "From"	100
4.2.1.8	Column "Subject"	100
4.2.1.9	Column "Received"	101
4.2.1.10	Column "Size"	101
4.2.1.11	Column "Categories"	101
4.2.2	PidTagViewDescriptorStrings	102
4.3	Conversation Actions	102
4.3.1	A Categorized and Moved Conversation Action	102
4.4	Navigation Shortcut	104
4.4.1	Group Header	104
4.4.2	Navigation Shortcut	105
5	Security	107
5.1	Security Considerations for Implementers	107
5.2	Index of Security Parameters	107
6	Appendix A: Office/Exchange Behavior	107
	Index	111

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.

1.1 Glossary

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

- ASCII**
- attende**
- Calendar object**
- code page**
- conversation action**
- Coordinated Universal Time (UTC)**
- E-mail object**
- EntryID**
- folder**
- folder associated information (FAI)**
- group header**
- GUID**
- handle**
- little-endian**
- Meeting Request object**
- Meeting Response object**
- message**
- Navigation shortcut.**
- non-Unicode**
- property**
- property ID**
- rule**
- special folder**
- store**
- Stream object**
- table**
- Unicode**
- XML**

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

- Boolean**
- Byte**
- ULONG**

WORD

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

PtypBinary
PtypBoolean
PtypCurrency
PtypFloating64
PtypFloatingTime
PtypErrorCode
PtypGuid
PtypInteger16
PtypInteger32
PtypErrorCode
PtypString
PtypString8
PtypTime

The following terms are specific to this document:

category list: A type of **configuration data** that contains a list of textual labels with associated data such as color. Other attributes of a category include a shortcut key that can be used to quickly apply a category, a usage counter, the last time the category was applied or used by the user, and a **GUID**.

configuration data: A group of application settings. Each group is uniquely identified by the **folder** that contains the group, the name of the group, and whether the group is contained in the **dictionary** or **XML** stream.

dictionary: A type of **configuration data** that consists of a table of name-value property pairs. Each setting has a unique name property within the table.

folder flags: A collection of **subproperties** on a **folder** that are stored together in a single **property**. This requires that all the **subproperties** are always read or written together.

subproperty: A binary stream property that is embedded in another **property**, possibly along with other **subproperties**.

view: A UI mechanism that displays a table of the **messages** in a **folder**.

view definition: A collection of parameters for a **view**. These parameters include the **display names** of the columns and the **properties** that they contain. These parameters include any of the following:

- The set of properties that the application uses to split the rows into groups with collapsible header rows on each group.
- The set of properties that the application uses to sub-sort the rows within any groups that are included.
- A restriction that the application uses to filter the set of rows in the table.

working hours: A type of **configuration data** that consists of structured data that describes the user's preferred working hour pattern. The structure includes the start time, end time, and days of the week of the user's working hours. To enable translation between time zones, the structure also includes a description of the user's home time zone, including the offset from **UTC** and any rules that describe a daylight saving time transition.

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

1.2 References

1.2.1 Normative References

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

[MS-OXCADATA] Microsoft Corporation, "Data Structures Protocol Specification", June 2008.

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

[MS-OXCICAL] Microsoft Corporation, "iCalendar to Appointment Object Conversion Protocol Specification", June 2008.

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

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

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

[MS-OXCRPC] Microsoft Corporation, "Wire Format Protocol Specification", June 2008.

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

[MS-OXGLOS] Microsoft Corporation, "Exchange Server Protocols Master Glossary", June 2008.

[MS-OXOABK] Microsoft Corporation, "Address Book Object Protocol Specification", June 2008.

[MS-OXOCAL] Microsoft Corporation, "Appointment and Meeting Object Protocol Specification", June 2008.

[MS-OXOFLAG] Microsoft Corporation, "Informational Flagging Protocol Specification", June 2008.

[MS-OXORMDR] Microsoft Corporation, "Reminder Settings Protocol Specification", June 2008.

[MS-OXORULE] Microsoft Corporation, "E-Mail Rules Protocol Specification", June 2008.

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

[MS-OXOSRCH] Microsoft Corporation, "Search Folder List Configuration Protocol Specification", June 2008.

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

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

[XMLBase] W3C, "XML Base", June 2001, <http://www.w3.org/TR/xmlbase/>.

1.2.2 Informative References

None.

1.3 Protocol Overview

Clients and servers might need to share some settings with one another. For example, if a user has configured a client with his preferred working hours, both the client and the server might use those settings to help choose optimal times for new appointments on the user's schedule.

Other settings can be used by the client to change the behavior of a server feature, and vice versa. An example of this would be when the server and clients have mutually exclusive features, and only one of them can be enabled at a time. Storing the state of the client feature in a shared location would allow the server to disable its corresponding feature, and vice versa.

The Configuration Information protocol specifies the settings that are shared between clients and servers, and the manner in which the settings are shared. 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**

In addition to the settings that are defined in the Configuration Information protocol, the client or server might 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 Configuration Information protocol when they are only used by a single application.

1.3.1 Configuration Data

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

The streams 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. Most simple settings use this type of stream.

For more structured data, such as the user's preferred working hours, the streams 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 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 and assign color values to those categories.

1.3.2 View Definitions

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

1.3.3 Folder Flags

Folder flags consist of a collection of small properties. These properties are packed into a single binary property on a **folder**. The primary purpose of the folder flags is to store Boolean flags that affect the way that the folder can be displayed.

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.

1.3.4 Conversation Actions

Conversation actions are a limited set of actions that a user applies to all **E-mail objects**, currently in the store or delivered in the future, that share the same **PidTagConversationId**. Unlike **rules** (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** can be a combination of any of the following actions:

- Ignore
- Move to folder
- Categorize

1.4 *Relationship to Other Protocols*

The Configuration Information protocol works with the Folder Object protocol [MS-OXCFOLD], the Message and Attachment Object protocol [MS-OXCMSG], the Special Folders protocol [MS-OXOSFLD], and the Table Object protocol [MS-OXCTABL]. Figure 1 shows the relationship between these protocols.

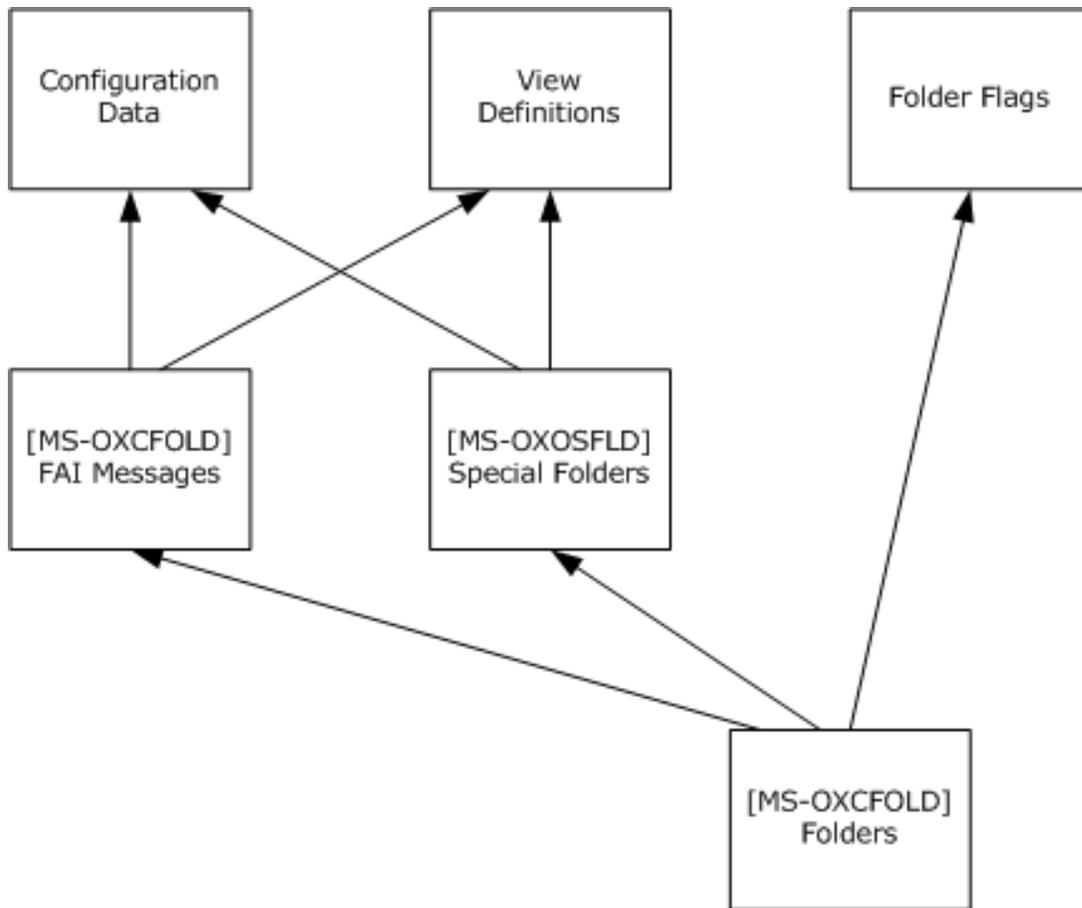


Figure 1: Protocol Stack

The **configuration data** and **view definitions** components of the Configuration Information protocol use **FAI messages** [MS-OXCFOLD] as the transport. The FAI messages are sometimes contained in a **special folder**; therefore, these components need to use the Special Folders protocol [MS-OXOSFLD].

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

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

This document specifies versioning issues in the following areas:

- Capability negotiation -The client first checks the version number returned by the server in the results from **EcDoConnectEx**, as specified in [MS-OXCRPC]. If the server returns a version that is greater than or equal to 14.0.324.0, the client never deletes the conversation action FAI message (section 3.1.4.7) and disables almost all processing of incoming E-mail objects (see section 3.1.5.1).

1.8 Vendor-Extensible Fields

A third-party application can store its own settings using **FAI** messages by specifying its own custom **PtypString** for the value of the **PidTagMessageClass PtypString** property. A centralized authority that ensures uniqueness of the **PidTagMessageClass PtypString** property across different applications does not exist.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The following sections specify how Configuration Information protocol **messages** use **properties** and streams [MS-OXCPRPT] that have been set on **FAI** messages or **folders** [MS-OXCFOLD] as the underlying transport.

2.2 Message Syntax

The following sections specify the location and format of the property and stream buffers that are specific to the Configuration Information protocol.

2.2.1 XML Format

The supported **XML** format to be read and written as **configuration data** in this protocol is a subset of the W3C recommendation [XMLBase].

Applications MUST NOT depend on support for namespaces [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
"	"
<	<
>	>
&	&

Applications SHOULD escape the following special characters within quoted strings: <1>

Special Character	Escape Sequence
'	'

2.2.2 Binary Format

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

2.2.3 Configuration Data

The client and server SHOULD store certain settings as **configuration data**<2>. The format and location of the configuration data, as well as which settings it can include, are defined in the following subsections.

The application stores configuration data in an **FAI message**. The application stores the FAI message in the **special folder** that is defined in the following sections for each type of configuration data.

The message MUST have the **PidTagMessageClass PtypString** property 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 PtypInteger32** property set. The value of the property MUST be a bitmask that indicates which stream properties exist on the

message. The streams types, and thus the flags, are not mutually exclusive. The bitmasks are as follows:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved																											a	b	c		

Reserved: These bits are unused. They SHOULD be set to 0. The client and server MUST ignore these bits if they are set.

a: If this bit is set, the FAI message SHOULD contain a **dictionary** stream, serialized into a fixed XML schema and stored in the **PidTagRoamingDictionary** stream property. These streams are defined in section 2.2.3.1. If the FAI message does not contain a dictionary stream, the application MUST treat the dictionary as having no entries.

b: If this bit is set, the FAI message contains an XML stream stored in the **PidTagRoamingXmlStream** stream property that uses an arbitrary XML schema. These streams are defined in section 2.2.3.2.

c: This bit is unused. It SHOULD be set to 0. The client and server MUST ignore this flag if it is set.

2.2.3.1 Dictionaries

A message with a **dictionary** stream MUST have the **PidTagRoamingDatatypes PtyInteger32** property set. The value of the property MUST be a bitmask that includes 0x00000004.

The message MUST have the **PidTagRoamingDictionary** stream property set. The value of the property is a binary stream that contains a **Unicode XML** document using the UTF8 encoding. The XML document MUST conform to the following XSD schema, in addition to the limitations specified in section 2.2.1.

```
<?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:annotation>
    <xs:documentation xml:lang="en-us">
      The name and version of the application that created this
      document can be encoded in the version string. There is
      no validation of this information, it is just provided for
      future reference. The format of the version string is:

      &lt;name&gt;.&lt;major version number&gt;
    </xs:documentation>
  </xs:annotation>
  <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:annotation>
    <xs:documentation xml:lang="en-us">
      Different value types are all encoded in this simpleType as a
      string. The format of the string is:

      &lt;data type&gt;-&lt;string encoded value&gt;
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{3,9}[-0-9A-Za-z]{0,255}" />
  </xs:restriction>
</xs:simpleType>

```

The data type is an integer type code from the following list:

- 3: Boolean
- 9: 32-bit signed integer
- 18: String (0 or more Unicode characters)

The encoding of the string encoded value depends on the data type:

- 3 (Boolean): "True" or "False"
- 9 (32-bit signed integer): Decimal characters, prefixed with an optional "-" to denote a negative number.

```

        18 (String): Unicode string
    </xs:documentation>
</xs:annotation>
<xs:restriction>
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="\d+\-.*" />
        </xs:restriction>
    </xs:simpleType>
</xs:restriction>
</xs:simpleType>
</xs:schema>

```

Info: A top-level 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** be encoded in the version string. The data is not validated; it is provided for future reference. The format of the version string is:

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

Data: A top-level element that **MUST** contain all the dictionary name-value pair entries.

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

k: An attribute on the **e** element 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 on the **e** element 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:

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

The data type **MUST** be an integer type code from the following list:

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": The client MUST prefer its own default or locally stored settings, and it MUST rewrite the XML document with those settings.
 - "1": The client MUST prefer the settings in the XML document.
 - Default: (32-bit integer) "0"

2.2.3.1.1 *Calendar Options*

If the client or server supports **configuration data**, it stores the settings defined 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**.

This dictionary MUST have the **PidTagMessageClass PtypString** property 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].
 - Default: (missing) The default behavior when this setting is missing is specified in [MS-OXORMDR].
- **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**, as specified in [MS-OXOCAL], 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 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"

2.2.3.2 XML Streams

The message **MUST** have the **PidTagRoamingDatatypes PtyInteger32** property set. The value of the property **MUST** be a bitmask that includes 0x00000002.

The message **MUST** have the **PidTagRoamingXmlStream** stream property set. The value of the property **MUST** be a **PtyBinary** stream that contains a **Unicode XML** document that is using the UTF8 encoding.

In addition to the XSD schemas that are specified in the following subsections, the XML document **MUST** conform to the limitations specified in section 2.2.1.

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.

2.2.3.2.1 Working Hours

If the client or server supports **configuration data**, it stores the settings that are defined in this section in a **Working Hours** stream. The application stores the Working Hours stream in an **FAI message** contained in the Calendar **special folder**.

The message **MUST** have the **PidTagMessageClass PtyString** property set. The value of the property **MUST** be "IPM.Configuration.WorkHours".

The **XML** document that is stored in **PidTagRoamingXmlStream** **MUST** conform to the following XSD schema.

```
<?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: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**, and **DaylightSavings** elements.

Bias: This element **MUST** exist and contains the offset in minutes of the user's current time zone from **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** <5> exist and contains the standard name of the time zone described by the data in **TimeZone**. See the **KeyName** field of the **PidLidAppointmentTimeZoneDefinitionRecur** property described in [MS-OXOCAL] for possible values.

DSTTransition: This is a **complexType** that 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** from the **DSTTransition** type **MUST** be added to the **Bias** contained in the **WorkHoursVersion1** element when

this transition takes effect, which **MUST** be determined by the **ChangeDate** element.

ChangeDate: This element determines when the transition takes place. The **Bias** specified in the **DSTTransition** **MUST** be added to the time zone bias after the transition. This element contains a **Time**, **Date**, and **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 defined in this section. If the year is any other value, the application performs the transition of 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 2nd Sunday of the month. The following are the possible values:

Value	Description
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 this 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 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 following are the possible values:

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

2.2.3.2.2 *Category List*

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

The message **MUST** have the **PidTagMessageClass PtypString** property set. The value of the property **MUST** be "IPM.Configuration.CategoryList".

The **XML** document that is stored in **PidTagRoamingXmlStream** **MUST** conform to the following XSD schema.

```
<?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:annotation>
      <xs:documentation>
        Only values 0-24 map to a color. Applications SHOULD use the
        value -1 for no color, any other value SHOULD also map to no
        color.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:int">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="keyboardShortcutType">
    <xs:annotation>
      <xs:documentation>
        Only values 1-11 map to a shortcut key. Applications SHOULD
        use the value 0 for no shortcut key, any other value SHOULD
        also map to no shortcut key.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:unsignedInt">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="dateTimeRestrictedType">
    <xs:annotation>
      <xs:documentation>
        dateTime type used in this XSD has the following additional
        restrictions:

        The year MUST be between 1601 and 30827.

        The time 24:00:00 is not valid.

        Fractional seconds SHOULD have 3 digit precision (i.e.
        milliseconds). The application MAY include additional
        digits. The application SHOULD handle any extra digits
        if they are included.

        The application MUST specify the time in UTC. The
        application MAY append a Z for the timezone identifier. The
        application MUST ignore any other timezone specifier and
```

```

        interpret the time using UTC.
    </xs:documentation>
</xs:annotation>
<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>

```

name: 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).

SHOULD NOT be in the form of the string representation of a **GUID**, as specified in the guid field in this section<6>.

color: 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. Applications can choose to display the color category differently.

Value	Base R,G,B	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

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: 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).
<7>

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

usageCount: Reserved. Applications SHOULD write 0 <8>.

lastTimeUsed, **lastTimeUsedMail**, **lastTimeUsedCalendar**, **lastTimeUsedContacts**, **lastTimeUsedTasks**, **lastTimeUsedNotes**, and **lastTimeUsedJournal**: See section 3.1.4.2.3 for a definition of these elements.

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 handle any extra digits if they are included<9>.

The application MUST specify the time in **UTC**. The application can append a **Z** for the time zone identifier. The application MUST ignore any other time zone identifier and interpret the time using UTC.

lastSessionUsed: Reserved. Applications SHOULD write 0 <10>.

guid: A GUID that identifies the category and does not change if the user renames the category. The GUID MUST be stored in a string in the form of "{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}" where X is any hexadecimal digit.

renameOnFirstUse: If set to "1", an application can prompt the user to rename the category when it is first applied to a message (as specified in section 3.1.4.6) <11>. If the user renames the category 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: The name of a category in the category list that is to be applied (as specified in section 3.1.4.6) if the application provides a one-click method to apply a category.

lastSavedSession: Reserved. Applications SHOULD write 0 <12>.

lastSavedTime: The value MUST be set to the time in UTC when the category list was saved.

2.2.4 View Definitions

The client and server SHOULD store certain settings as **view definitions**<13>. The format and location of the view definitions, as well as which settings are included, are defined in the following subsections.

A **message** that contains view definitions MUST be an **FAI** message. 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 PtypString** property set and the value of the property is "IPM.Microsoft.FolderDesign.NamedView".

The message has the **PidTagViewDescriptorVersion PtypInteger32** property set and the value of the property is 0x00000008.

The message has the **PidTagViewDescriptorName PtypString** property set and the value of the property is a non-empty string.

The view definitions MUST be stored as a binary stream in the stream property **PidTagViewDescriptorBinary** of the message. The column headers are stored in the **PidTagViewDescriptorStrings** stream property on the message as **non-Unicode** strings using the current **code page** of the client. The following sections specify the packet format of these two properties respectively.

2.2.4.1 PidTagViewDescriptorBinary

View definitions MUST be stored in stream property **PidTagViewDescriptorBinary** of the message. It is in binary format and the packet structure is specified as follows:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved1																															
...																															
Version																															
ulFlags																															
Pres																															
Cvcd																															
IvcdSort																															
cCat																															
ulCatSort																															
Reserved2 (24 bytes)																															
...																															
...																															
...																															
...																															
...																															
ColumnInfo (variable)																															
RestrictionInfo (optional, variable)																															

Reserved1

Size: 8 bytes

This field **MUST** exist. The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

Version

Data type: **ULONG**

This field **MUST** exist. This is a fixed value of 0x00000008.

ulFlags

Data type: **ULONG**

This field **MUST** exist. This specifies the sort order of the sorted column. The value of this field **MUST** be one of the following:

Value	Meaning
0x00000000	Ascending sort order
0x00000002	Descending sort order

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

Pres

Data type: **ULONG**

This field **MUST** exist. This field is filled with arbitrary value by client and **SHOULD NOT** be used by server.

Cvcd

Data type: **ULONG**

This field **MUST** exist. It specifies the number of **ColumnInfo** fields that are stored in this packet.

ivcdSort

Data type: **ULONG**

This field **MUST** exist. The value of this field **MUST** be one of the following:

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

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

cCat

Data type: **ULONG**

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 **Cvcd**, whichever is least.

ulCatSort

Data type: **ULONG**

This field **MUST** exist. This field uses bit flags to specify the ascending or descending order of the "group by" columns. The flags are defined as follows. In each case, if the flag is not set, the "group by" column is in descending order.

Flag	Description
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

Size: 24 bytes

This field **MUST** exist. The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

ColumnInfo

Data type: **ColumnPacket** structure, as specified in section 2.2.4.1.1.

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:

1. The "blank" column: This is a single column that **MUST** have the following settings, as defined in section 2.2.4.1.3:

Field	Value
Vcids	0x00040001
Cx	0x00000007
Flags	0x00000028 (VCDF_BITMAP VCDF_NOT_SORTABLE)
Kind	0x00000000
IID	0x00000004

2. 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.
3. The "visible" columns: All columns that **MUST** be visible to users excluding the "group by" columns.
4. The "order by" column: If the sorted column is not a "group by" or "visible" column, it **MUST** be stored here.

RestrictionInfo

Data type: **RestrictionPacket** structure, as specified in section 2.2.4.1.2.

This is where the restriction of the table view **MUST** be stored.

2.2.4.1.1 ColumnPacket

The ColumnPacket packet **MUST** contain the information of a single column including the property identifier, property type, and display attributes. The structure of the packet **MUST** be as follows:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
PropertyType																PropertyID															
Cx																															
Reserved1																															
Flags																															
Reserved2																															
...																															
...																															
Kind																															
ID																															
Guid (optional)																															
BufferLength (optional)																															
Buffer (optional, variable)...																															

PropertyType

Data type: **WORD**

This field **MUST** exist. This field specifies the type of the property, as specified in section 2.2.4.1.3.1.

PropertyID

Data type: **WORD**

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**, the value **MUST** be truncated and the two least significant bytes **MUST** be stored in this field.

Cx

Data type: **ULONG**

This field **MUST** exist. This specifies the column width in pixels.

Reserved1

Size: 4 bytes

This field **MUST** exist. The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

Flags

Data type: **ULONG**

This field **MUST** exist. This field contains column descriptor flags. The bit setting and its meaning are listed in the following table.

Name	Value	Description
VCDF_LEFT_JUSTIFY	0x00000000	Column is left justified.
VCDF_RIGHT_JUSTIFY	0x00000001	Column is right justified.
VCDF_CENTER_JUSTIFY	0x00000002	Column is center justified.
VCDF_BITMAP	0x00000008	Column header is in bitmap format.
VCDF_NOT_SORTABLE	0x00000020	Column is not sortable.
VCDF_SORTDESCENDING	0x00000040	Column is sorted in descending order.
VCDF_MOVEABLE	0x00000100	Deprecated.
VCDF_COLUMNSDLG	0x00000200	Deprecated.
VCDF_SORTDLG	0x00000400	Column MUST be able to be sorted.
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	Deprecated.
VCDF_MULTIVALUED	0x00004000	Indicates whether the column PropertyType field MUST include the MV_FLAG flag.

Reserved2

Size: 12 bytes

This field **MUST** exist. The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

Kind

Data type: **ULONG**

This field **MUST** exist. The field contains one of the following values:

Name	Value	Description
KindID	0x00000000	The property uses an integer identifier.
KindString	0x00000001	The property uses a string identifier.

ID

Data type: **ULONG**

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 **RopGetPropertyIdsFromNames** [MS-OXCROPS] 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 and **MUST** ignore the value of this field when reading the stream.

Guid

Data type: **GUID**

If the **VCDF_NAMEDPROP** flag is set in the **Flags** field, this field contains the GUID that **MUST** be used with **RopGetPropertyIdsFromNames** to translate the named property into a **property ID**. If the **VCDF_NAMEDPROP** flag is not set, the application **MUST** omit this field.

BufferLength

Data type: **ULONG**

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 NULL terminator character (0x0000). Otherwise, the application **MUST** omit this field.

Buffer

Data type: **Unicode String**

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 **RopGetPropertyIdsFromNames** to translate the named property into a **property ID**. Otherwise, the application **MUST** omit this field. This field includes a Unicode NULL terminator character (0x0000) at the end.

2.2.4.1.2 RestrictionPacket

Restrictions are used to evaluate the content table of the folder. Only those rows with **TRUE** result of the evaluation **MUST** be displayed.

The restrictions are stored using a special format, which is different from the format specified for restrictions in [MS-OXCDATA]. The packet starts from a single **RestrictionBlock** buffer. A **RestrictionBlock** buffer **MUST** consist of a restriction type, a restriction data record, a number that indicates property tag and value pairs, and a list of the property tag and value pair. When the restriction type indicates a compositional condition, for example **AND** or **OR**, more restriction blocks **MUST** follow after the current restriction block. A restriction is a packet recursively built up by **RestrictionBlock**. To determine the size of the restriction, the application parses each **RestrictionBlock** recursively if necessary.



RestrictionBlock

Data type: **RestrictionBlock** structure, as specified in 2.2.4.1.3.

This field contains the restriction type. From the restriction type it can be determined whether it contains subrestrictions and the number of subrestrictions that it contains. The server parses each **RestrictionBlock** recursively, if necessary, to complete reading one restriction block.

2.2.4.1.3 RestrictionBlock

Each restriction block MUST contain the type of condition. Based on the type, the data record, property tags, and property values can be determined. If the type is AND, OR, NOT, SubObject, and Comment, more subrestrictions MUST follow this restriction block.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
RestrictionData																															
PropValueNum (optional)																															
PropValue (optional)...																															
...PropValue (optional)...																															
...PropValue (optional)																															
PropValue (optional)...																															
...PropValue (optional)...																															
...PropValue (optional)																															
PropValue (optional)...																															
...																															

RestrictionType

Data type: **ULONG**

This field specifies which condition is in use. The following table specifies all available restriction types.

Name	Value	Description
RES_AND	0x00000000	Specifies a Logical AND condition.
RES_OR	0x00000001	Specifies a Logical OR condition.
RES_NOT	0x00000002	Specifies a Logical NOT condition.
RES_CONTENT	0x00000003	Content condition.
RES_PROPERTY	0x00000004	Specifies a Property condition.
RES_COMPAREPROPS	0x00000005	Specifies a Compare Properties condition.

RES_BITMASK	0x00000006	Specifies a Bit Mask condition.
RES_SIZE	0x00000007	Specifies a Size condition.
RES_EXIST	0x00000008	Specifies an Exist condition.
RES_SUBRESTRICTION	0x00000009	Specifies a Sub Object condition.
RES_COMMENT	0x0000000A	Specifies a Comment condition.

RestrictionData

Size: 12 Bytes

This field specifies the actual data record that is associated with the restriction type. The content of this structure varies based on the restriction type. Each restriction type and its corresponding data structure is specified in the following sections.

PropValueNum

Data type: **ULONG**

This field can be present, depending on the restriction type. If it is present, it specifies the number of **PropValues** that follow. See the following sections for details. This field SHOULD NOT exist unless otherwise specified for each restriction type.

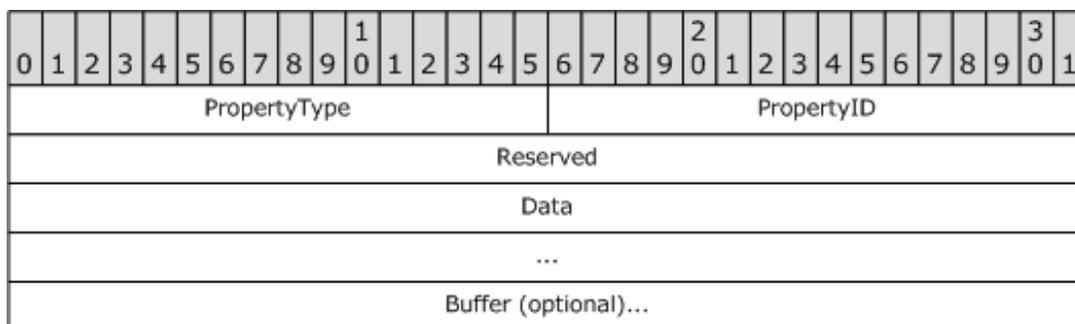
PropValue

Data type: **PropValue** structure, as specified in section 2.2.4.1.3.1.

The **PropValue** field can be present, depending on the restriction type. If it is present, this field MUST appear the number of times specified in the **PropValueNum** field. This field SHOULD NOT exist unless otherwise specified for each restriction type.

2.2.4.1.3.1 PropValue

Each **RestrictionBlock** can contain one or more **PropValue** fields. The **PropValue** field MUST use the following format:



PropertyType

Data Type: **WORD**

This field specifies the type of the property. The property type **MUST** be one of the following:

Type Name	Type ID
PtypInteger16	0x0002
PtypInteger32	0x0003
PtypFloating32	0x0004
PtypFloating64	0x0005
PtypCurrency	0x0006
PtypFloatingTime	0x0007
PtypErrorCode	0x000A
PtypBoolean	0x000B
PtypInteger64	0x0014
PtypString8	0x001E
PtypTime	0x0040
PtypGuid	0x0048
PtypBinary	0x0102

PropertyID

Data Type: **WORD**

This field specifies the ID of the property.

Reserved

Size: 4 bytes

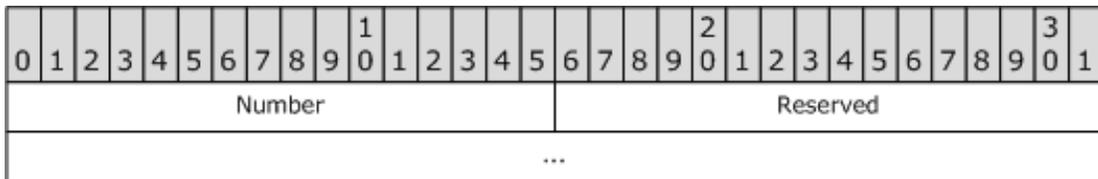
The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

Data

Size: 8 bytes

The format of this field depends on the property type specified in the **PropertyType** field, as follows:

PtypInteger16: The **Data** field **MUST** contain a 2-byte signed integer:



Number

Size: 2 bytes

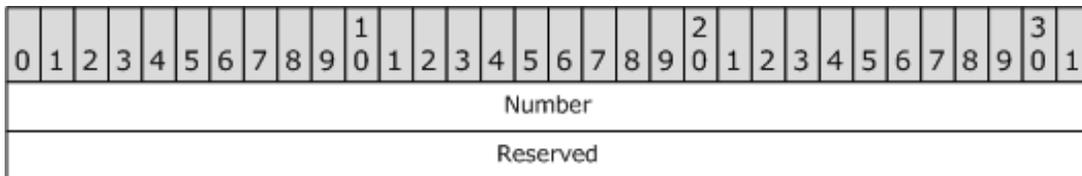
This field contains a signed integer.

Reserved

Size: 6 bytes

The application can fill this field with values of 0 when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

PtypInteger32: The **Data** field **MUST** contain a 4-byte signed integer:



Number

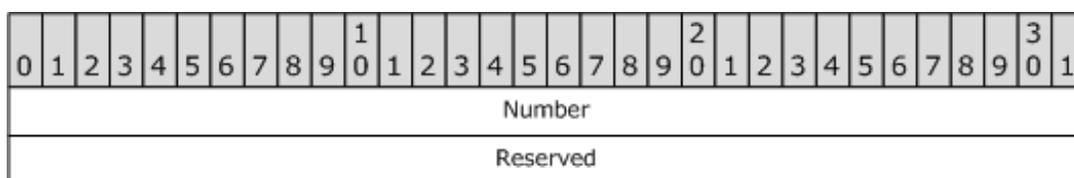
Size: 4 bytes

This field contains a signed integer.

Reserved

Size: 4 bytes

The application can fill this field with values of 0 when writing the stream. The application MUST ignore the value of this field when reading the stream.

PtypFloating32: The **Data** field MUST contain a 4-byte floating point number:**Number**

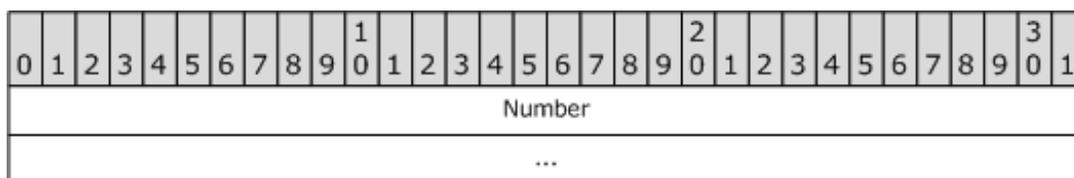
Size: 4 bytes

This field contains a floating point number.

Reserved

Size: 4 bytes

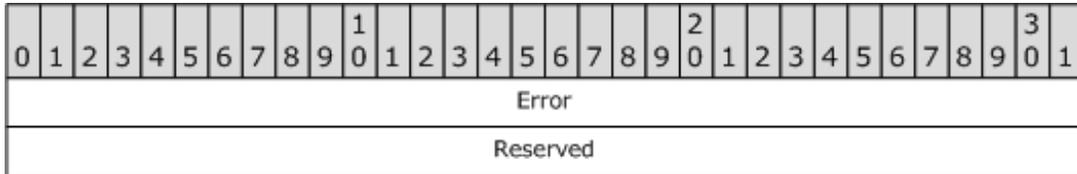
The application can fill this field with values of 0 when writing the stream. The application MUST ignore the value of this field when reading the stream.

PtypFloating64: See **PtypFloatingTime** in this section.**PtypCurrency:** See **PtypInteger64** in this section.**PtypFloatingTime:** The **Data** field MUST contain an 8-byte floating point number:**Number**

Size: 8 bytes

This field contains a floating point number.

PtypErrorCode: The **Data** field MUST contain a 4-byte SCODE error code:



Error

Size: 4 bytes

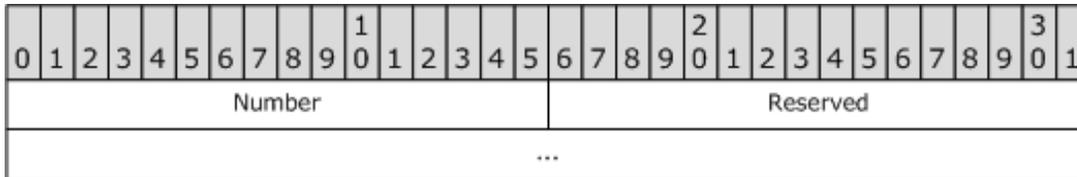
This field contains an SCODE error code.

Reserved

Size: 4 bytes

The application can fill this field with values of 0 when writing the stream.
The application MUST ignore the value of this field when reading the stream.

PtypBoolean: The **Data** field MUST contain a **WORD**:



Number

Size: 2 bytes

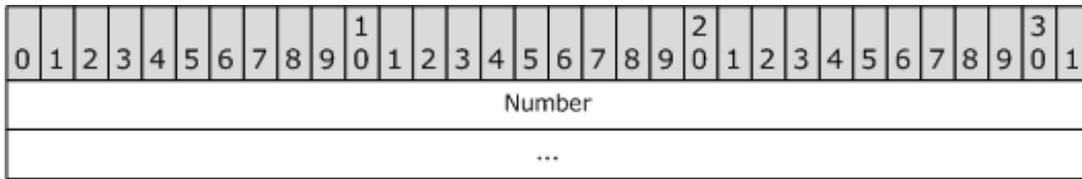
This field contains an unsigned integer.

Reserved

Size: 6 bytes

The application can fill this field with values of 0 when writing the stream.
The application MUST ignore the value of this field when reading the stream.

PtypInteger64: The **Data** field MUST contain an 8-byte signed integer:



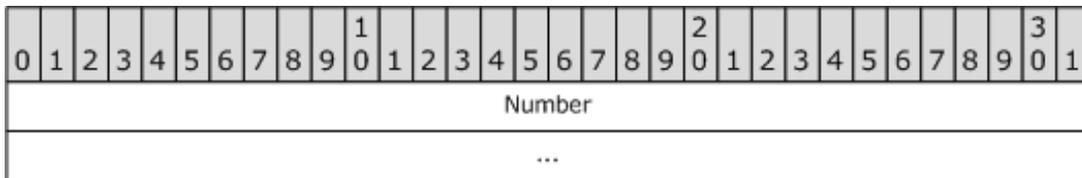
Number

Size: 8 bytes

This field contains a signed integer.

PtypString8: See **PtypGuid** in this section. The application **MUST** store the string value separately in **PidTagViewDescriptorStrings**, as specified in section 2.2.4.2.

PtypTime: The **Data** field **MUST** contain an 8-byte unsigned integer:

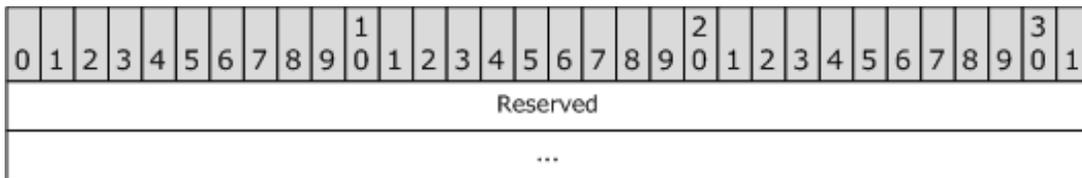


Number

Size: 8 bytes

This field contains an unsigned integer.

PtypGuid: The **Data** field is reserved:

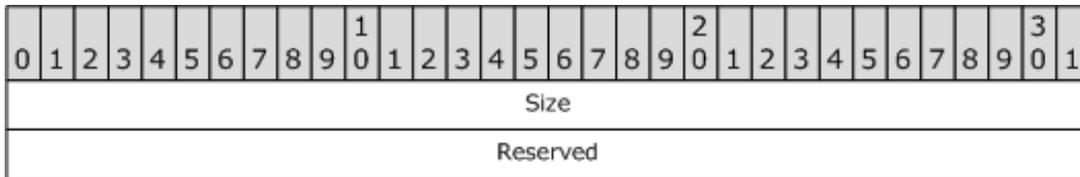


Reserved

Size: 8 bytes

The application can fill this field with values of 0 when writing the stream.
The application **MUST** ignore the value of this field when reading the stream.

PtypBinary: The **Data** field **MUST** contain a 4-byte unsigned integer:



Size

Size: 4 bytes

This field contains an unsigned integer.

Reserved

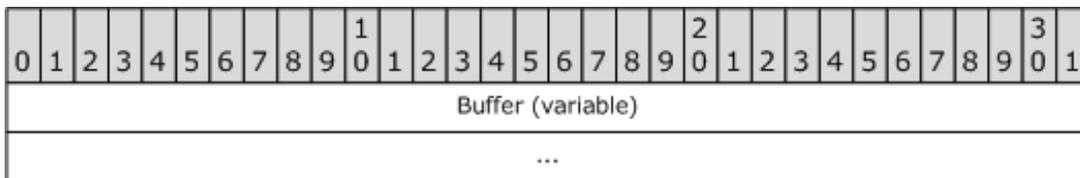
Size: 4 bytes

The application can fill this field with values of 0 when writing the stream.
The application **MUST** ignore the value of this field when reading the stream.

Buffer

Size: Variable length

This field **MUST** exist only when the property type encoded in **PropertyType** is **PtypBinary**. The **Buffer** field contains an arbitrary binary stream. The size of the stream is specified as a number of bytes in the **Size** field within **Data**.



2.2.4.1.3.2 Logical AND Condition

The Logical **AND** condition is used to join a group of two or more conditions by using a logical **AND** operation. The result of the **AND** condition is **TRUE** if all of the child conditions evaluate to **TRUE**. The result is **FALSE** if any child condition evaluates to **FALSE**.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
cRes																															
Reserved																															
...																															
SubCondition...																															
...																															
...SubCondition																															
SubCondition...																															
...																															

RestrictionType: RES_AND

cRes

Data type: **ULONG**

Specifies the number of conditions that make up the **AND** condition. Each subcondition is stored in a **RestrictionBlock** and all subconditions are stored sequentially in the packet.

Reserved

Size: 8 bytes

The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

SubCondition

Data type: **RestrictionBlock**, as specified in 2.2.4.1.3.

This field specifies subconditions that make up the AND condition.

2.2.4.1.3.3 Logical OR Condition

The Logical **OR** condition is used to join a group of two or more conditions by using a logical **OR** operation. The result of the **OR** condition is **TRUE** if any of the child conditions evaluates to **TRUE**. The result is **FALSE** if all child conditions evaluate to **FALSE**.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
cRes																															
Reserved																															
...																															
SubCondition...																															
...																															
...SubCondition																															
SubCondition...																															
...																															

RestrictionType: RES_OR

cRes

Data type: **ULONG**

It specifies the number of conditions that make up the **OR** condition.

Reserved

Size: 8 bytes

The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

SubCondition

Data type: **RestrictionBlock**, as specified in 2.2.4.1.3.

This field specifies subconditions that make up the **OR** condition.

2.2.4.1.3.4 Logical NOT Condition

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

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
Reserved																															
...																															
...																															
SubCondition...																															
...																															
...SubCondition																															

RestrictionType: RES_NOT

Reserved

Size: 12 bytes

The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

SubCondition

Data type: **RestrictionBlock**, as specified in 2.2.4.1.3.

It specifies a single subcondition that makes up the **NOT** condition.

2.2.4.1.3.5 Content Condition

The Content condition is used to search properties that have contents that match a search string.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
ulFuzzyLevel																															
PropertyType																PropertyID															
Reserved																															
PropValueNum																															
PropValue																															

RestrictionType: RES_CONTENT

ulFuzzyLevel:

Data type: **ULONG**

This field specifies flags that control the behavior of the string comparisons that are used to evaluate the restriction. The lower 16 bits of the fuzzy level are mutually exclusive:

Name	Values	Description
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 following values, and can be combined using the logical **OR** operation.

Name	Values	Description
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.

PropertyType

Data type: **WORD**

This field specifies the type of the property, as specified in section 2.2.4.1.3.1.

PropertyID

Data type: **WORD**

This field specifies the ID of the property, as specified in section 2.2.4.1.3.1.

Reserved

Size: 4 bytes

The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

PropValueNum

Data type: **ULONG**

This field is specified in section 2.2.4.1.3. This field **MUST** exist in this type of restriction, and the value **MUST** be 0x00000001.

PropValue

Data type: **PropValue** structure, as specified in section 2.2.4.1.32.1.

This field appears exactly once in this type of restriction.

2.2.4.1.3.6 Property Condition

The Property condition is used to compare the value of a property with a constant.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RestrictionType																															
RelOp																															
PropertyType																PropertyID															
Reserved																															
PropValueNum																															
PropValue																															

RestrictionType: RES_PROPERTY

RelOp

Data type: **ULONG**

This field specifies the relational operator to be used in the search. The value **MUST** be one of the following:

Name	Values	Description
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

Data type: **WORD**

This field specifies the type of the property, as specified in section 2.2.4.1.3.1.

PropertyID

Data type: **WORD**

This field specifies the ID of the property, as specified in section 2.2.4.1.3.1.

Reserved

Size: 4 bytes

The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

PropValueNum

Data type: **ULONG**

This field is specified in section 2.2.4.1.3. This field **MUST** exist in this type of restriction, and the value **MUST** be 0x00000001.

PropValue

Data type: **PropValue** structure, as specified in section 2.2.4.1.3.1.

This field appears exactly once in this type of restriction.

2.2.4.1.3.7 Compare Properties Condition

The Compare Properties condition is used to describe a condition that tests two properties by using a relational operator.

The Property condition is used to compare the value of a property with a constant.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
RelOp																															
PropertyType1																PropertyID1															
PropertyType2																PropertyID2															

RestrictionType: RES_COMPAREPROPS

RelOp

Data type: **ULONG**

This field specifies the relational operator that is to be used in the search, as specified in section 2.2.4.1.3.6.

PropertyType1

Data type: **WORD**

This field specifies the type of the first property, as specified in section 2.2.4.1.3.1.

PropertyID1

Data type: **WORD**

This field specifies the ID of the first property, as specified in section 2.2.4.1.3.1.

PropertyType2

Data type: **WORD**

This field specifies the type of the second property, as specified in section 2.2.4.1.3.1.
The type of the second property **MUST** match the type of the first property.

PropertyID2

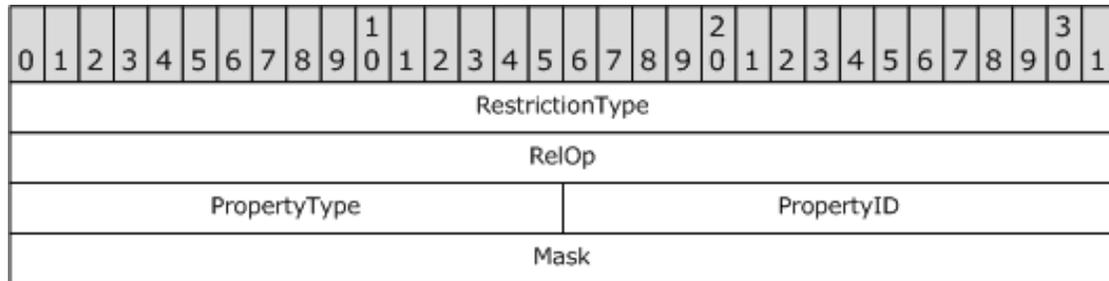
Data type: **WORD**

This field specifies the ID of the second property, as specified in section 2.2.4.1.3.1.

2.2.4.1.3.8 Bit Mask Condition

The Bit Mask condition is used to perform a bitwise **AND** operation on the value of the property and to test the result produced by the operation.

The **RestrictionBlock** of this restriction type **MUST** use the following format:



RestrictionType: RES_BITMASK

RelOp

Data type: **ULONG**

This field specifies the relational operator that is to be used in the search. The value **MUST** be one of the following:

Name	Value	Description
BMR_EQZ	0x00	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	0x01	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

Data type: **WORD**

This field specifies the type of the property, as specified in section 2.2.4.1.3.1.

PropertyID

Data type: **WORD**

This field specifies the ID of the property, as specified in section 2.2.4.1.3.1.

Mask

Data type: **ULONG**

This field specifies the bitmask that the application **MUST** use in a bitwise **AND** with the value of the property when performing the search.

2.2.4.1.3.9 Size Condition

The Size condition is used to test the size of a property value.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

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

RestrictionType: RES_SIZE

RelOp

Data type: **ULONG**

This field specifies the relational operator that is to be used in the search, as specified in section 2.2.4.1.32.6.

PropertyType

Data type: **WORD**

This field specifies the type of the property, as specified in section 2.2.4.1.3.1.

PropertyID

Data type: **WORD**

This field specifies the ID of the property, as specified in section 2.2.4.1.3.1.

Size

Data type: **ULONG**

This field specifies the size in bytes that **MUST** be compared with the size of the value of this property when performing the search.

2.2.4.1.3.10 Exist Condition

The Exist condition is used to test whether a particular property exists on a **message**.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
Reserved1																															
PropertyType																PropertyID															
Reserved2																															

RestrictionType: RES_EXIST

Reserved1

Size: 4 bytes

The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

PropertyType

Data type: **WORD**

This field specifies the type of the property, as specified in section 2.2.4.1.3.1.

PropertyID

Data type: **WORD**

This field specifies the ID of the property, as specified in section 2.2.4.1.3.1.

Reserved2

Size: 4 bytes

The application can fill this field with any value when writing the stream. The application MUST ignore the value of this field when reading the stream.

2.2.4.1.3.11 SubObject Condition

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

The **RestrictionBlock** of this restriction type MUST use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
RestrictionType																															
SubObject																															
Reserved																															
...																															
SubCondition...																															
...																															
...SubCondition																															

RestrictionType: RES_SUBRESTRICTION

SubObject

Data type: **ULONG**

The application MUST use one of the following values for this field:

Value	Description
0xE1200D	Apply the condition to the recipient table of a message.
0xE1300D	Apply the condition to the attachment table of a message.

Reserved

Size: 8 bytes

The application can fill this field with any value when writing the stream. The application MUST ignore the value of this field when reading the stream.

SubCondition

Data type: **RestrictionBlock**, as specified in 2.2.4.1.3.

This field specifies a single subcondition that makes up the SubObject condition.

2.2.4.1.3.12 Comment Condition

Comment conditions are unlike other conditions because the conditions are not evaluated, but are only used for reference by the application. The comment condition is used to keep additional application-specific information with the restriction in the form of an arbitrary list of property tag and value pairs.

The **RestrictionBlock** of this restriction type **MUST** use the following format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
RestrictionType																																	
cValues																																	
Reserved																																	
...																																	
PropValueNum																																	
PropValue...																																	
...																																	
...PropValue																																	
PropValue...																																	
...																																	
...PropValue																																	
...																																	
SubCondition...																																	
...																																	
...SubCondition																																	

RestrictionType: RES_COMMENT

cValues

Data type: **ULONG**

This field **MUST** have the same value as the **PropValueNum** field.

Reserved

Size: 8 bytes

The application can fill this field with any value when writing the stream. The application **MUST** ignore the value of this field when reading the stream.

PropValueNum

Data type: **ULONG**

This field is specified in section 2.2.4.1.3. This field **MUST** exist in this type of restriction.

PropValue

Data type: **PropValue** structure, as specified in section 2.2.4.1.3.1.

This field occurs the number of times specified in the **PropValueNum** field, as specified in section 2.2.4.1.3.1.

SubCondition

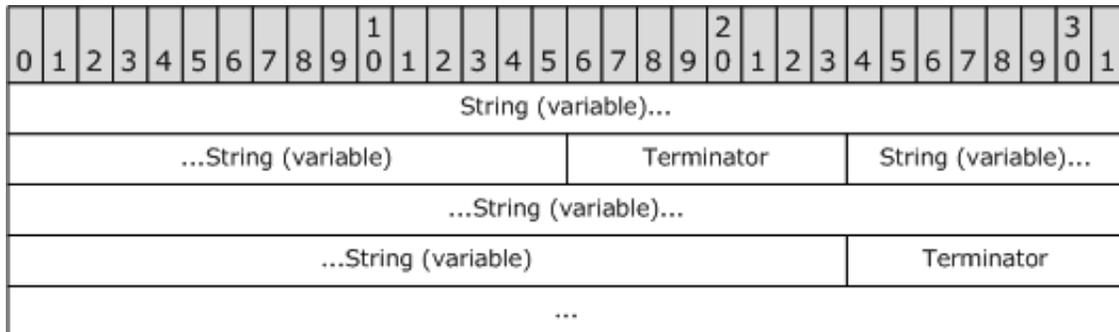
Data type: **RestrictionBlock**, as specified in 2.2.4.1.1.

This field specifies a single subcondition that makes up the comment condition.

2.2.4.2 PidTagViewDescriptorStrings

The client **MUST** store the display strings referenced in **PidTagViewDescriptorBinary** separately in the **PidTagViewDescriptorStrings** property. The client **MUST** concatenate the strings in the same order in which the strings are referenced in **PidTagViewDescriptorBinary**. The first set of strings consists of the display names of each of the **ColumnInfo** structures, followed by the value of each **PropValue** structure that uses the **PtypString8** property type.

The client **MUST** use the following binary layout in **PidTagViewDescriptorStrings**:



String

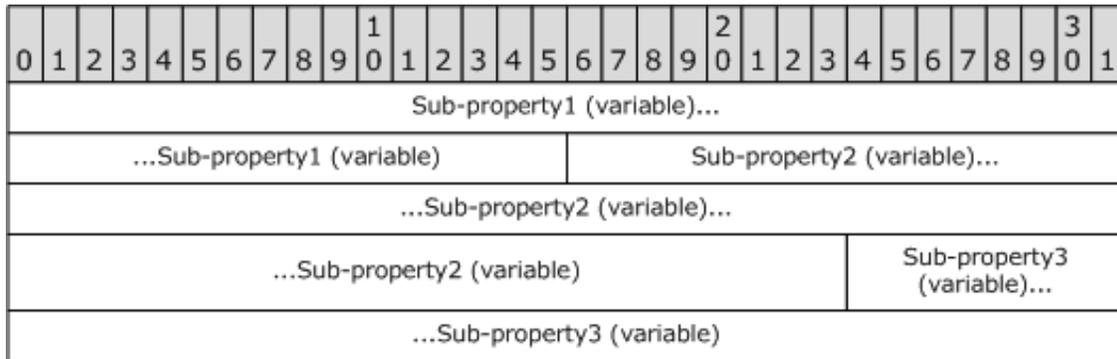
This field is an arbitrary length buffer that specifies the **non-Unicode** string by using the current **code page** of the client. The application **MUST NOT** include the byte value 0x0A, which corresponds to the **ASCII** newline character, in the **String**.

Terminator

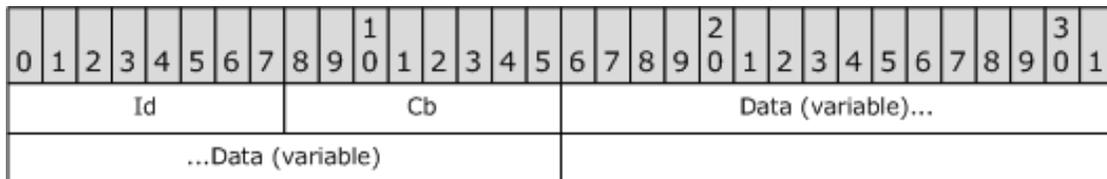
This field is a single byte that contains the value 0x0A, which corresponds to the ASCII newline character. The application **MUST** include a **Terminator** after every **String**, including the last **String** in the stream.

2.2.5 Folder Flags

The **PidTagExtendedFolderFlags** stream property can be set on a **folder**. If the property is set, the value of this property **MUST** be a binary stream that contains encoded sub-properties for the folder. The format of the binary stream **MUST** be as follows:



The binary stream is divided into variable-length **subproperty** fields. The subproperty fields are byte-aligned within the binary stream. Each subproperty **MUST** be encoded in the following format:



Id

The subproperty ID value. This field uses a fixed size of 1 byte. 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** 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 sub-property **Id** MUST appear 0 - 1 times in **PidTagExtendedFolderFlags**. The subproperties can appear in any order within the **PidTagExtendedFolderFlags** stream.

Name	Value
Invalid	0x00
ExtendedFlags	0x01
SearchFolderID	0x02
SearchFolderTag, as specified in [MS-OXOSRCH]	0x03
Reserved	0x04
ToDoFolderVersion	0x05
Reserved	0x06

Cb

This field specifies the unsigned size in bytes of the **Data** buffer of the subproperty. This field uses a fixed size of 1 byte.

Data

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 meaning of the subproperty depends on the value of the **Id** field. The **Id** field SHOULD have one of the following values:

Invalid

This value is invalid. The application MUST NOT use it.

ExtendedFlags

The **Data** field is a 4 byte field consisting of 32 bit flags. If the subproperty does not exist, or if the **PidTagExtendedFolderFlags** property is not set on the folder, each flag SHOULD assume the specified default value. The layout of the flags in the buffer MUST be as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
r1		a	r2			b	r3																								

r1

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

If the folder is subject to an administrative retention policy, this flag controls whether the application displays a string that describes the policy.

0: The application SHOULD <14> display a policy description..

1: The application MUST NOT display a policy description.

Default: 0

r2

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

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**.

00: The application uses the default value for this folder.

2.2.6.1 PidLidConversationActionLastAppliedTime

Type: **PtypTime**

This property SHOULD <17> 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.

2.2.6.2 PidLidConversationActionMaxDeliveryTime

Type: **PtypTime**

This property SHOULD<18> be the maximum value of PidTagMessageDeliveryTime 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).

2.2.6.3 PidLidConversationActionMoveFolderEid

Type: **PtypBinary**

This property SHOULD<19> be set based on the action(s) being performed on the conversation:

Action on the conversation	Value
Move to a folder in the same store as this FAI message	The entry ID of that folder
Move to a folder in a different store than this FAI message	The entry ID of that folder
Ignore	Byte array of size zero
None of the above	Not set

2.2.6.4 PidLidConversationActionMoveStoreEid

Type: **PtypBinary**

This property SHOULD<20> be set based on the action(s) being performed on the conversation:

Action on the conversation	Value
----------------------------	-------

Specifies the ID of the **Navigation shortcut** which groups other Navigation shortcuts. This property SHOULD only be set when the value of **PidTagWlinkType** is **wblHeader**.

2.2.7.4 PidTagWlinkSaveStamp

Type: **PtypInteger32**

Specifies an integer that allows a client to identify with a high probability whether the **Navigation shortcut** was saved by the current client session. When writing a Navigation shortcut, the client writes the Session ID (described in section 3.1.3.1) as the value for this property.

2.2.7.5 PidTagWlinkType

Type: **PtypInteger32**

Specifies the type of Navigation shortcut

Value	Name	Description
0x00000000	wblNormalFolder	The shortcut points to a folder other than those listed below.
0x00000001	wblSearchFolder	The shortcut points to a Search Folder
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.

2.2.7.6 PidTagWlinkFlags

Type :**PtypInteger32**

A bit field in which each bit SHOULD be set to 1 if the associated condition in the table below applies and 0 otherwise. The possible bit values are as follows.

Value	Name	Description
0x00000001	sipPublicFolder	Shortcut points to a server Public Folder
0x00000004	sipImapFolder	Shortcut points to a folder accessed via IMAP
0x00000008	sipWebDavFolder	Shortcut points to a folder accessed via WebDAV

0x00000010	sipSharePointFolder	Shortcut points to a folder accessed via SharePoint
0x00000020	sipRootFolder	Shortcut points to a root folder (IPM Subtree)
0x00000100	sipSharedOut	Shortcut points to a folder that has been shared out to other users. This value can be used to display a visual indicator and does not actually 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 actually set any permissions.
0x00000400	sipPersonFolder	Shortcut points to a folder belonging 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 PidTagWlinkOrdinal . Merging is purely a display – no objects are created or moved between folders.
0x00002000	sipOneOffName	Shortcut has been renamed such that the value of PidTagNormalizedSubject of the shortcut does not match the value of PidTagNormalizedSubject 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.

2.2.7.7 PidTagWlinkOrdinal

Type: **PtypBinary**

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.

2.2.7.8 PidTagWlinkEntryId

Type: **PtypBinary**

Specifies the **EntryID** of the folder pointed to by the shortcut.

2.2.7.9 PidTagWlinkRecordKey

Type: **PtypBinary**

Specifies the value of **PidTagRecordKey** of the folder pointed to by the shortcut.

2.2.7.10 PidTagWlinkStoreEntryId

Type: **PtypBinary**

Specifies the value of **PidTagStoreEntryId** of the folder pointed to by the shortcut.

2.2.7.11 PidTagWlinkFolderType

Type: **PtypGuid**

Specifies the type of folder pointed to by the shortcut. The possible values are as follows:

Value	Name	Description
0C78060000000000 C000000000000046	CLSID_MailFolder	The folder is a mail folder
0278060000000000 C000000000000046	CLSID_CalendarFolder	The folder is a calendar folder
0178060000000000 C000000000000046	CLSID_ContactFolder	The folder is a contact folder
0378060000000000 C000000000000046	CLSID_TaskFolder	The folder is a task folder
0478060000000000 C000000000000046	CLSID_NoteFolder	The folder is a note folder

0878060000000000 C000000000000046	CLSID_JournalFolder	The folder is a journal folder
--------------------------------------	---------------------	--------------------------------

2.2.7.12 PidTagWlinkGroupClsid

Type: **PtypGuid**

Specifies the value of **PidTagWlinkGroupHeaderID** of the **group header** associated with the shortcut.

2.2.7.13 PidTagWlinkGroupName

Type: **PtypString**

Specifies the value of **PidTagNormalizedSubject** of the **group header** associated with the shortcut.

2.2.7.14 PidTagWlinkSection

Type: **PtypInteger32**

Specifies the section where the shortcut should be grouped. The possible values are as follows:

Value	Name	Description
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

2.2.7.15 PidTagWlinkCalendarColor

Type: **PtypInteger32**

Specifies the background color of the calendar. The RGB values listed below 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

2.2.7.16 PidTagWlinkAddressBookEID

Type: **PtypBinary**

Specifies the value of **PidTagEntryId** of the user that the folder belongs to (see [MS-OXOABK]). This property SHOULD<30> be set on calendar shortcuts.

2.2.7.17 PidTagWlinkAddressBookStoreEID

Type: **PtypBinary**

Specifies the value of **PidTagStoreEntryId** of the current user (not the owner of the folder). This property SHOULD <31> be set on calendar shortcuts.

2.2.7.18 **PidTagWlinkClientID**

Type: **PtypGuid**

Specifies the Client ID that allows the client to determine whether the shortcut was created on the current machine/user via an equality test. The ID is specified in section 3.1.3.1. If this property is set, then the client compares the value of this property to the Client ID and only display the shortcut if the **GUIDS** match<32>.

2.2.7.19 **PidTagWlinkROGroupType**

Type: **PtypInteger32**

Specifies the type of **group header**. If the property does not exist, the client should assume a value of -1<33>. The possible values are as follows.

Value	Name	Description
-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

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is

provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.1.2 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.

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 `PtypGuid`, and is a randomly generated value.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Reading Configuration Data

To read settings in a **configuration data** settings group, the application **MUST** open the **special folder** that contains the configuration data **message**, as defined in [MS-OXOSFLD]. The application **MUST** call **RopGetContentsTable** with the `AssociatedFlag` flag to open the **FAI Contents Table**, as defined in [MS-OXCFOLD].

The application **MUST** find all the rows in the **FAI Contents Table** that have the **PidTagMessageClass PtypString** property specified by the configuration data message that the application is trying to open, by using steps equivalent to the following, as specified in [MS-OXCTABL]:

- Send **RopSetColumns** with a column set that includes the following properties:
 - **PidTagFolderId**
 - **PidTagMid**
 - **PidTagMessageClass**
 - **PidTagRoamingDatatypes**
- Send **RopSortTable** with a sort order that includes the following properties:
 - **PidTagMessageClass**, followed by
 - **PidTagLastModificationTime**
- Send **RopFindRow**, searching for a match on the **PidTagMessageClass PtypString** property.

- Send **RopQueryRows** repeatedly until either the end of the table or a row with a **PidTagMessageClass PtypString** property that no longer matches the configuration data message is encountered.
- Based on the subsort by **PidTagLastModificationTime**, pick the message with the most recent (the greatest value) modification time that includes the bit that matches the **PidTagRoamingDatatypes PtypInteger32** property specified by the configuration data message. If none of the messages match the **PidTagRoamingDatatypes PtypInteger32** property of the configuration data message, the application **MUST** pick the most recently modified of all the messages.

If the application 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 application **MUST** use default values for those settings when the configuration data message cannot be found.

If the application found a matching row, it **MUST** open the existing message by using steps equivalent to the following, as specified in [MS-OXCMSG]:

- Send **RopOpenMessage** to open the message, using the **PidTagFolderId** and **PidTagMid** properties from the table row and setting the ReadWrite flag in the *OpenModeFlags* parameter.

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

- Send **RopOpenStream** to open a **Stream object handle** on the stream property specified by the configuration data message.
- Read the serialized settings by using **RopReadStream**.

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 message that was picked above **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, an application **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 2.2.3.1.

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 2.2.3.1.

If the XML document does exist, and the XML document includes a valid **OLPrefsVersion** setting specified in section 2.2.3.1, 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, the **OLPrefsVersion** settings did not exist or was incorrect, 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 3.1.4.2. If any existing settings are found, the client **MUST** parse the **XML** document, as specified in section 2.2.3.2.

If the client could not find a matching configuration data message and it used default values<34>, or if the user changes the preferred working hours through the client UI, the client **MUST** generate the XML document as specified in section 2.2.3.2 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 data [MS-OXOCAL], 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** [MS-OXOSFLD] (because the other user's 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** is saved as an **FAI** message in the user's default Calendar **folder** [MS-OXOSFLD]. An application can choose to read the category list at any time.

When applications 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 is applied to a message (specified in section 3.1.4.6), or the user-visible properties of the category are changed (such as color or

shortcut key), the application SHOULD update the lastTimeUsed timestamp, and depending on whether the application separates different message types, the appropriate timestamp (Mail, Calendar, Contacts, Tasks, Notes, Journal), with the current time.

When viewing the contents of another user's folders [MS-OXOCAL], 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** [MS-OXOSFLD], 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 store is inaccessible or the client has not been granted sufficient permissions to access the special folder), the client MUST fall back to use its own category list settings.

3.1.4.2 Writing Configuration Data

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

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

- Send **RopOpenStream** to open a **stream object handle** on the stream property specified by the configuration data message.
- Read the serialized settings by using **RopReadStream**.

If the message does not exist, the application MUST create the message by using steps equivalent to the following, as specified in [MS-OXCMSG]:

- Send **RopCreateMessage** on the folder, passing the AssociatedFlag flag.

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

- Send **RopOpenStream** to open a **Stream object handle** on the stream property specified by the configuration data message.
- Write the serialized settings using **RopWriteStream**.

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 2.2.3.1.

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 2.2.3.2.1.

The client **MUST** generate the XML document as specified in section 2.2.3.2 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 2.2.3.2.2.

When applications 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 2.2.3.2 and save it to the configuration data message as specified in section 3.1.4.1.

Each category in the category list contains the following timestamps:

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

The lastTimeUsed timestamp **MUST** be set on all categories. All others are optional, and depend on whether the application shows different message types in separate windows or panes.

All times **MUST** be stored relative to **UTC**. When a category is applied to a message (specified in section 3.1.4.6), or the user visible properties of the category are changed (such as color or shortcut key), the application **SHOULD** update the lastTimeUsed timestamp, and depending on whether the application separates different message types, the appropriate timestamp (Mail, Calendar, Contacts, Tasks, Notes, Journal), with the current time.

3.1.4.3 Writing View Definitions

To write settings in a **view definition**, 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 **RopGetContentsTable** with the AssociatedFlag flag to open the **FAI Contents Table**, as defined in [MS-OXCFOLD].

If a View Definition message already exists with the same **PidTagViewDescriptorName** 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]:

- Send **RopSetColumns** with a column set that includes the following properties:
 - **PidTagFolderId**
 - **PidTagMid**
 - **PidTagMessageClass**
 - **PidTagViewDescriptorVersion**
 - **PidTagViewDescriptorName**
- Send **RopSortTable** with a sort order that includes the following properties:
 - **PidTagMessageClass**, followed by
 - **PidTagViewDescriptorVersion**
 - **PidTagViewDescriptorName**
- Send **RopFindRow**, searching for a match on the **PidTagMessageClass**, **PidTagViewDescriptorVersion**, and **PidTagViewDescriptorName** properties as defined in section 2.2.4.
- Send **RopQueryRows** 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 by using steps equivalent to the following, as specified in [MS-OXCFOLD]:

- Send **RopCreateMessage** on the folder, passing the AssociatedFlag flag.

If the client found a matching row, it MUST open the existing message using steps equivalent to the following, as specified in [MS-OXCFOLD]:

- Send **RopOpenMessage** to open the message, using the **PidTagFolderId** and **PidTagMid** properties from the table row and setting the ReadWrite flag in the *OpenModeFlags* parameter.

If the client found a matching message or created a new one, it **MUST** save the serialized settings streams on the properties specified in section 2.2.4, by using steps equivalent to the following, as specified in [MS-OXCFLD]:

- Send **RopOpenStream** to open **Stream object handles** on the **PidTagViewDescriptorBinary** and **PidTagViewDescriptorStrings** properties.
- Write the serialized settings by using **RopWriteStream**.

3.1.4.4 Reading Folder Flags

Reading and writing each of the **subproperties** in the **folder flags** are triggered by different events.

3.1.4.4.1 Reading ExtendedFolderFlags

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

3.1.4.4.2 Reading SearchFolderID

The client **MUST** read this value from every Search **folder** [MS-OXOSRCH] in the Finders **special folder** [MS-OXOSFLD]. Any search folder that has this **subproperty** is a persistent Search folder, and the client **SHOULD** display the search folder as such in the UI.

3.1.4.4.3 Reading ToDoFolderVersion

The client **MUST** read this value from the To-Do Search **folder** [MS-OXOSFLD] before it displays the contents of that folder. If the To-Do Search folder does not exist, it does not contain this **subproperty**, or it does not contain the required value as defined in section 2.2.5, the client **MUST** recreate the To-Do Search folder or reset the criteria of the Search folder, as specified in [MS-OXOFLAG].

3.1.4.5 Writing Folder Flags

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 2.2.5.

3.1.4.5.1 Writing ExtendedFolderFlags

Any time the user changes one of the display options for this **folder**, the client **MUST** re-write the **subproperty** to the folder.

3.1.4.5.2 Writing SearchFolderID

The client **MUST** write this **subproperty** on any new persistent Search **Folders** that it creates.

3.1.4.5.3 Writing ToDoFolderVersion

When the client recreates or resets the criteria on the To-Do Search **folder**, it MUST set this **subproperty** on the folder.

3.1.4.6 Applying a category to a Message

A **message** can have a list of categories stored in the **PidNameKeywords** multi-value **PtypMultipleString** property. To apply a new category to a message, the application MUST read the current value of **PidNameKeywords** from the message and check to see if the current value already contains the name of the new category. If the current value does not include the name of the new category, the application MUST insert the name of the category in the list and set the new value of **PidNameKeywords** on the message.

3.1.4.7 Performing a Conversation Action

Certain user actions performed on a conversation SHOULD modify the **conversation actions** on a conversation and trigger special handling logic:

- Ignore
- Unignore
- Move to folder
- Stop moving to folder
- 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 **PidTagConversationId** 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 **PidTagConversationIndex** (see section 2.2.6.72.2.6.7) corresponding to the given conversation ID. If no such **FAI message** exists, proceed to step 4.
2. If the user is performing an Unignore action and the **FAI message** does not indicate that there is an Ignore **conversation action** (as specified in sections 2.2.6.3 and 2.2.6.4), the client SHOULD NOT perform the Unignore action for this conversation ID.
3. If the user is performing a Stop moving to folder action and the **FAI message** does not indicate that there is a Move to folder **conversation action** (as specified in sections 2.2.6.3 and 2.2.6.4), the client SHOULD NOT perform the Stop moving to folder 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 2.2.6

5. The client SHOULD locate all **E-mail objects** on the store with a **PidTagConversationId** 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 on all located **E-mail objects**, as specified in the table below:

User Action	Action to perform on each E-mail 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 .
Unignore	If the E-mail object is in the Deleted Items special folder , move the E-mail object to the Inbox special folder .
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.
Stop moving to folder	None
Add category	Append the user-specified category to the PidNameKeywords property of each E-mail object
Remove category	Remove the user-specified category from the PidNameKeywords property of each E-mail object . 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

7. The client SHOULD set the latest value of **PidTagMessageDeliveryTime** from the located **E-mail objects** as **PidLidConversationActionMaxDeliveryTime** on the **FAI message**. If no messages were found, **PidLidConversationActionMaxDeliveryTime** SHOULD be set to 00:00:00 (UTC) April 1, 1601.
8. The client SHOULD set the current time (in UTC) as **PidLidConversationActionLastAppliedTime** on the **FAI message**.
9. The client SHOULD perform the appropriate action on the **FAI message**, as specified in the table below:

User Action	Action to perform on the FAI message object
Ignore	Set PidLidConversationActionMoveFolderEid to an array of size 0. Delete PidLidConversationActionMoveStoreEid .

Unignore	Delete PidLidConversationActionMoveFolderEid and PidLidConversationActionMoveStoreEid .
Move to folder	Set PidLidConversationActionMoveFolderEid to the entry ID of the user-specified folder. If the user-specified folder is in the same store as the FAI message , delete the PidLidConversationActionMoveStoreEid . Otherwise, set PidLidConversationActionMoveStoreEid to the entry ID of the user-specified store.
Stop moving to folder	Delete PidLidConversationActionMoveFolderEid and PidLidConversationActionMoveStoreEid .
Add category	Append the user-specified category to the PidNameKeywords property of the FAI message .
Remove category	Remove the user-specified category 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 the FAI message .

10. If the server returned a version less than 14.0.324.0 (as detailed in section 1.7.1) and **PidLidConversationActionMoveFolderEid** and **PidNameKeywords** are both not set, the client SHOULD delete the **FAI message**.

3.1.4.8 Reading Navigation Shortcuts

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

3.1.4.9 Writing Navigation Shortcuts

3.1.4.9.1 Group Headers

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

- **PidTagWlinkGroupHeaderID**
- **PidTagWlinkSaveStamp**
- **PidTagWlinkType**

- PidTagWlinkFlags
- PidTagWlinkOrdinal
- PidTagWlinkFolderType
- PidTagWlinkSection

3.1.4.9.2 Shortcuts

Shortcuts have the following properties:

- PidTagWlinkSaveStamp
- PidTagWlinkType
- PidTagWlinkFlags
- PidTagWlinkOrdinal
- PidTagWlinkEntryId
- PidTagWlinkRecordKey
- PidTagWlinkStoreEntryId
- PidTagWlinkFolderType
- PidTagWlinkGroupClsid
- PidTagWlinkGroupName
- PidTagWlinkSection

Shortcuts with the value of **PidTagWlinkSection** set to wbsidCalendar SHOULD <37> have the following properties if needed or appropriate:

- PidTagWlinkCalendarColor
- PidTagWlinkAddressBookEID
- PidTagWlinkAddressBookStoreEID
- PidTagWlinkROGroupType

Shortcuts have the following available to limit the visibility of a shortcut to a single machine:

- PidTagWlinkClientID

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 store, the client SHOULD<38> perform the following actions:

1. The client SHOULD locate the **conversation action FAI message** in the Conversation Actions Settings **special folder** with bytes 6-21 of PidTagConversationIndex (see section 2.2.6.7) corresponding to the PidTagConversationId 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 **PidLidConversationProcessed** 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 on the incoming **E-mail object** based on the properties of the **FAI message**, as specified below (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
PidNameKeywords is set	If the server returned a version less than 14.0.324.0 (as detailed in section 1.7.1), the categories in the FAI message SHOULD be appended to PidNameKeywords of the incoming E-mail object .
PidLidConversationActionMoveStoreEid is not set and PidLidConversationActionMoveFolderEid is not an array of size zero	If the server returned a version less than 14.0.324.0 (as detailed in section 1.7.1), then the client SHOULD move the incoming E-mail object to the folder with an entry ID equal to the PidLidConversationActionMoveFolderEid on the FAI message .
PidLidConversationActionMoveStoreEid is not set and PidLidConversationActionMoveFolderEid is an array of size zero	If the server returned a version less than 14.0.324.0 (as detailed in section 1.7.1), then the client SHOULD move the incoming E-mail object to the Deleted Items special folder .
PidLidConversationActionMoveStoreEid is set	(Note that this case is not conditioned on the server version) The client SHOULD move the incoming E-mail object to the folder and store identified by the PidLidConversationActionMoveFolderEid and PidLidConversationActionMoveStoreEid, respectively.

4. If the server returned a version less than 14.0.324.0 (as detailed in section 1.7.1) and if step 3 did not encounter an error opening the store, PidLidConversationProcessed SHOULD be set to a random number on the incoming **E-mail object**.
5. The client SHOULD set the current time (in UTC) as PidLidConversationActionLastAppliedTime on the **FAI message**.

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** on an incoming **E-mail object** simultaneously. If the server returned a version less than 14.0.324.0 (as detailed in section 1.7.1), the client SHOULD<39> perform duplicate detection on incoming **E-mail objects**:

1. The client SHOULD compare PidTagSearchKey property on the incoming **E-mail object** to the **PidTagSearchKey** of all recently delivered E-mail objects in the same folder<40>.
2. If a matching **E-mail object** was found, the client SHOULD compare the unsigned values of the PidLidConversationProcessed property 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<41> perform additional processing on the copy of the **E-mail object** in the Sent Items **special folder**.

1. The client SHOULD locate the conversation action **FAI message** in the Conversation Actions Settings **special folder** with bytes 6-21 of PidTagConversationIndex (see section 2.2.6.7) corresponding to the PidTagConversationId 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**.
2. If PidNameKeywords is set on the **FAI message**, the categories in the **FAI message** SHOULD be appended to PidNameKeywords 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 detailed in section 1.7.1) and the elapsed time between the PidLidConversationActionLastAppliedTime of the **FAI message** and the current time is greater than an application-specific duration<42>, the client SHOULD<43> delete the **FAI message**.

3.1.7 Other Local Events

None.

3.2 Server Details

Clients operate on **folders** and **messages** using the [MS-OXPROPS] protocol. 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

The Abstract Data Model for the server and client roles are the same. See section 3.1.1.

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** 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 **PidTagCreationTime** 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** [MS-OXOSFLD]. If the client tries to access the **configuration data message** without the necessary permissions, the server **MUST** deny access to the message.

3.2.4.1.2 Reading Category List

The server can place limits on the size of the **XML** document that it will parse<44>.

The server is responsible for enforcing permissions that the user grants to the Calendar **special folder** [MS-OXOSFLD]. If the client tries to access the **configuration data message** without the necessary permissions, the server **MUST** deny access to the message.

3.2.4.2 Writing Configuration Data

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 server **SHOULD** pick the message with the earliest modification time stored in **PidTagLastModificationTime** 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**, the server **MUST** enumerate all of the View Definition **FAI messages** in the folders, searching for a match on the **PidTagMessageClass** and **PidTagViewDescriptorVersion** properties as defined in section 2.2.4.

After the server has built the list of View Definition messages, it can select one of them by using the **PidTagViewDescriptorName** property. After it has selected a View Definition message, the server **MUST** read the settings from the **PidTagViewDescriptorBinary** and **PidTagViewDescriptorStrings** properties on the message.

3.2.4.4 Reading Folder Flags

Reading and writing each of the **subproperties** in the **folder flags** are triggered by different events.

3.2.4.4.1 Reading ExtendedFolderFlags

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

3.2.4.4.2 Reading SearchFolderID

The server **MUST** read this value from every Search **folder** [MS-OXOSRCH] in the Finders **special folder** [MS-OXOSFLD]. Any Search folder that has this **subproperty** is a persistent Search folder, and the server **SHOULD** display the Search folder as such in the UI.

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 2.2.5.

3.2.4.5.1 Writing ExtendedFolderFlags

Any time the user changes one of the display options for this **folder**, the server **MUST** re-write the **subproperty** to the folder.

3.2.4.5.2 Writing ToDoFolderVersion

When the server recreates or resets the criteria [MS-OXOFLAG] on the To-Do Search **folder** [MS-OXOSFLD], it **MUST** set this **subproperty** on the folder.

3.2.4.6 Applying a category to a Message

The server handles this event the same way that the client handles it, as specified in section 3.1.4.6.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Processing a Change to a Conversation Action FAI Message

When the client uploads a change to a **conversation action FAI message**, the server SHOULD<45>:

1. Locate all **E-mail objects** in the store that have a PidTagConversationId matching bytes 6-21 of the PidTagConversationIndex (see section 2.2.6.7) on the **FAI message** and that have a PidTagMessageDeliveryTime greater than the PidLidConversationActionMaxDeliveryTime on the **FAI message**.
2. Perform the appropriate action on all located **E-mail objects**, based on the changes to the **FAI message**, as specified in the table below (more than one action can apply to a given change):

FAI message change	Action to apply on the E-mail object
PidLidConversationActionMoveFolderEid 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 .
PidLidConversationActionMoveFolderEid 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 special folder .
PidLidConversationActionMoveFolderEid changed to a non-empty array	If PidLidConversationActionMoveStoreEid 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 entry ID equal to the new PidLidConversationActionMoveFolderEid on the FAI message .
PidLidConversationActionMoveFolderEid was a non-empty array, now unset	None
PidNameKeywords unset, set, or modified	For all categories added in this change, append the category to the PidNameKeywords property of each E-mail object . For all categories removed in this change, remove the category 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 store, the server SHOULD<46>:

1. Locate the **conversation action FAI message** in the Conversation Actions settings **special folder** with bytes 6-21 of PidTagConversationIndex (see section 2.2.6.7) corresponding to the PidTagConversationId 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 on the incoming **E-mail object** based on the properties of the **FAI message**, as specified below (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
PidNameKeywords is set	The categories in the FAI message SHOULD be appended to PidNameKeywords of the incoming E-mail object .
PidLidConversationActionMoveStoreEid is not set and PidLidConversationActionMoveFolderEid is not an array of size zero	The server SHOULD move the incoming E-mail object to the folder with an entry ID equal to the PidLidConversationActionMoveFolderEid on the FAI message .
PidLidConversationActionMoveStoreEid is not set and PidLidConversationActionMoveFolderEid is an array of size zero	The server SHOULD move the incoming E-mail object to the Deleted Items special folder .

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<47> perform additional processing on the copy of the **E-mail object** in the Sent Items **special folder**:

1. The server SHOULD locate the conversation action **FAI message** in the Conversation Actions settings **special folder** with bytes 6-21 of PidTagConversationIndex (see section 2.2.6.7) corresponding to the **PidTagConversationId** 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 **PidNameKeywords** is set on the **FAI message**, the categories in the **FAI message** SHOULD be appended to **PidNameKeywords** 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 on a **configuration data message**, as specified in section 2.2.3.1:

```
<?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 on a **configuration data message**, as specified in section 2.2.3.2.1:

```
<?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>
    </TimeZone>
  </WorkHoursVersion1>
</Root>
```

```

        </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 on a **configuration data message**, as specified in section 2.2.3.2.2:

```

<?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 stores the column description data and the **PidTagViewDescriptorStrings** property stores the column headers.

4.2.1 PidTagViewDescriptorBinary

The following is the complete buffer:

```

0000: 00 00 00 00 00 00 00 00 00-08 00 00 00 02 00 00 00
0010: 00 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 00
0030: 00 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 00-28 00 00 00 00 00 00 00
0050: 00 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 00 4A 2F 00 00
0070: 00 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 00
0090: 40 3F 00 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 00

```

```

00b0: C0 00 00 00 00 00 00 00 46-1E 00 1A 00 12 00 00 00
00c0: 00 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 00-1A 00 00 00 03 00 90 10
00e0: 12 00 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 00 90 10 00 00
0100: 0B 00 1B 0E 12 00 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 00
0120: 1B 0E 00 00 1E 00 42 00 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 00
0140: 00 00 00 00 42 00 00 00 00-1E 00 37 00 11 00 00 00
0150: 00 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 00-37 00 00 00 40 00 06 0E
0170: 10 00 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 00 06 0E 00 00
0190: 03 00 08 0E 0C 00 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 00
01b0: 08 0E 00 00 1E 10 00 00 00-12 00 00 00 00 00 00 00
01c0: 20 7B 00 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 00
01e0: C0 00 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 00 00

```

The first 8 bytes are reserved:

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

The next four bytes specify **Version**. After Version it is the ulFlags value:

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

Version: 0x00000008

ulFlags: 0x00000002 (VDF_SORTDESCENDING)

This **ulFlags** 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

cvcd = 0x0B 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: 0

This means that the view is sorted by column "Received" and the sort order 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: 0

Because the table is not grouped, this 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 restrictions, 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

Vcds: 0x00040001

0030: 01 00 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

IID: 0x00000004

0050: 04 00 00 00

4.2.1.2 Column "Importance"

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

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

Vcbs: 0x00170003, or **PidTagImportance**

0060: 03 00 17 00

Cx: 0x00000012

0060: 12 00 00 00

Reserved1:

0060: 00 00 00 00

Flags: 0x0004A2F, or (VCDF_BITMAP | VCDF_CENTER_JUSTIFY |
VCDF_SORTDLG | VCDF_GROUPDLG | VCDF_SORTDESCENDING |
VCDF_RCOLUMNSDLG | VCDF_MOVEABLE | VCDF_COLUMNSDLG)

0060: 4A 2F 00 00

Reserved2:

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

Kind: 0x00000000

0070: 00 00 00 00

IID: 0x00000017

0080: 17 00 00 00

4.2.1.3 Column "Reminder"

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

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

Vcids: 0x8503000B, or PidLidReminderSet

0080: 0B 00 03 85

Cx: 0x00000012

0080: 12 00 00 00

Reserved1:

0080: 00 00 00 00

Flags: 0x0003F40, or (VCDF_NAMEDPROP | VCDF_SORTDESCENDING | VCDF-RCOLUMNSDLG | VCDF_SORTDLG | VCDF_GROUPDLG | VCDF_MOVEABLE)

0090: 40 3F 00 00

Reserved2:

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

Kind: 0x00000000

00a0: 00 00 00 00

IID: 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 column "Icon":

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

Vcids: 0x001A001E, or PidTagMessageClass

Cx: 0x00000012

Flags: 0x000270A

Kind: 0x00000000

IID: 0x0000001A

4.2.1.5 Column "Flag Status"

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

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

Vcids: 0x10900003, or PidTagFlagStatus

Cx: 0x00000012

Flags: 0x0002F4A

Kind: 0x00000000

IID: 0x00001090

4.2.1.6 Column "Attachment"

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

```
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
```

Vcbs: 0x0E1B000B, or **PidTagHasAttachments**

Cx: 0x00000012

Flags: 0x0002F4A

Kind: 0x00000000

IID: 0x00000E1B

4.2.1.7 Column "From"

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

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

Vcbs: 0x0042001E, or **PidTagSentRepresentingName**

Cx: 0x0000000C

Flags: 0x0002F00

Kind: 0x00000000

IID: 0x00000042

4.2.1.8 Column "Subject"

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

```
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
0160: 00 00 00 00 00 00 00 00-37 00 00 00
```

Vcbs: 0x0037001E, or **PidTagSubject**

Cx: 0x00000011

Flags: 0x0002F00

Kind: 0x00000000

IID: 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
```

Vcids: 0x0E060040, or **PidTagMessageDeliveryTime**

Cx: 0x00000010

Flags: 0x0002F40

Kind: 0x00000000

IID: 0x00000E06

4.2.1.10 Column "Size"

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

```
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
```

Vcids: 0x0E080003, or **PidTagMessageSize**

Cx: 0x0000000C

Flags: 0x0002740

Kind: 0x00000000

IID: 0x00000E08

4.2.1.11 Column "Categories"

Next in the buffer is the description of column "Categories". This is a column with named property **PidNameKeywords**:

```
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 00
```

Vcids: 0x0000101E, or **PidNameKeywords**
Cx: 0x00000012
Flags: 0x0007B20
Kind: 0x00000001
Guid: {00020329-0000-0000-C000-000000000046}
BufferLength: 0x00000012
Buffer: "Keywords"

4.2.2 PidTagViewDescriptorStrings

In this example, **PidTagViewDescriptorStrings** contains all the column headers delimited by '\n':

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

4.3 Conversation Actions

4.3.1 A Categorized and Moved Conversation Action

Suppose that 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 **PidTagConversationId**:

```
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 "Fabrikam" and "Business Proposals" to the conversation. At 3:51 PM, the user then moves the conversation to a folder, "FY09 Archive", with an **entry ID** of:

```
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
```

in a store, "Archived Mail", with an **entry ID** of:

```
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	Value
PidLidConversationActionLastAppliedTime	23:51:11 2/17/2009
PidLidConversationActionMaxDeliveryTime	23:31:42 2/17/2009
PidLidConversationActionMoveFolderEid	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	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
PidLidConversationActionVersion	0x003CCCCC
PidNameKeywords	{"Fabrikam", "Business Proposals"}

PidTagConversationIndex	0000: 01 00 00 00 00 00 00 B7 A2- B5 C4 AA 65 1C F2 D3 8C 0000: 62 8C 0E AF 56 C4
PidTagMessageClass	"IPM.ConversationAction"
PidTagSubject	"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 the Calendar section.

The client sends a **RopCreateMessage** request with the FolderId set to the Id of the Common Views folder and the Associated Flag set to 1, and 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** request to set the following properties:

Property	PropertyID	Property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	"IPM.Microsoft.WunderBar.Link"
PidTagNormalizedSubject	0x0E1D	0x001F (PtypString)	"My Work Calendars"
PidTagWlinkGroupHeaderID	0x6842	0x0102 (PtypBinary)	5BA943D8DAAA462CA63E9136F65C8681
PidTagWlinkSaveStamp	0x6847	0x0003 (PtypInteger32)	0x12345678
PidTagWlinkType	0x6849	0x0003 (PtypInteger32)	0x00000004

PidTagWlinkFlags	0x684A	0x0003 (PtypInteger32)	0x00000000
PidTagWlinkOrdinal	0x684B	0x0102 (PtypBinary)	0x80
PidTagWlinkFolderType	0x684F	0x0102 (PtypBinary)	0278060000000000 C0000000000000046
PidTagWlinkSection	0x6852	0x0003 (PtypInteger32)	0x00000003

The client then sends a **RopSaveChangesMessage** request to persist the object on the server, and a **RopRelease** request 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 section 4.4.1.

The client sends a **RopCreateMessage** request with the FolderId set to the Id of the Common Views folder and the Associated Flag set to 1, and waits for the server to respond. The server response contains a handle to the **Message object**.

The client then sends a **RopSetProperties** request to set the following properties:

Property	PropertyID	Property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	"IPM.Microsoft.WunderBar.Link"
PidTagNormalizedSubject	0x0E1D	0x001F (PtypString)	"Meetings"
PidTagWlinkSaveStamp	0x6847	0x0003 (PtypInteger32)	0x12345678
PidTagWlinkType	0x6849	0x0003 (PtypInteger32)	0x00000000

PidTagWlinkFlags	0x684A	0x0003 (PtypInteger32)	0x00000000
PidTagWlinkOrdinal	0x684B	0x0102 (PtypBinary)	0x80
PidTagWlinkEntryId	0x684C	0x0102 (PtypBinary)	See below
PidTagWlinkRecordKey	0x684D	0x0102 (PtypBinary)	See below
PidTagWlinkStoreEntryId	0x684E	0x0102 (PtypBinary)	See below
PidTagWlinkFolderType	0x684F	0x0102 (PtypBinary)	0278060000000000 C000000000000046
PidTagWlinkGroupClsid	0x6850	0x0102 (PtypBinary)	5BA943D8DAAA462C A63E9136F65C8681
PidTagWlinkGroupName	0x6851	0x001F (PtypString)	"My Work Calendars"
PidTagWlinkSection	0x6852	0x0003 (PtypInteger32)	0x00000003

The value of **PidTagWlinkEntryId**, **PidTagWlinkRecordKey**, and **PidTagWlinkStoreEntryId** are copied directly from the corresponding properties of the actual calendar folder as described in sections 2.2.7.8, 2.2.7.9, and 2.2.7.10.

The client then sends a **RopSaveChangesMessage** request to persist the object on the server, and a **RopRelease** request 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 (for details, see [MS-OXCMMSG]).

5.2 Index of Security Parameters

None.

6 Appendix A: Office/Exchange Behavior

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

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

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

<1> Section 2.2: In Outlook 2007 and Outlook 2010 Cached Mode, when the local cached copy of the **Category List** is in conflict with the copy on the server, Outlook 2007 and Outlook 2010 sometimes does not escape the single quote character.

<2> Section 2.2.3.1: Outlook 2003 and Exchange 2003 do not read or write **Configuration Data FAI** messages.

<3> Section 2.2.3.1: Exchange 2007 uses the string “Exchange.12” and 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.

<4> Section 2.2.3.1.1: Outlook 2003 and Outlook 2007 do not support the **piAutoDeleteReceipts** property.

<5> Section 2.2.3.2.1: Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 do not read nor write this element.

<6> Section 2.2.3.2.2: This restriction on the category name is supported only by Exchange 2003.

<7> Section 2.2.3.2.2: Exchange 2003 do not use this attribute data.

<8> Section 2.2.3.2.2: Outlook 2007 and Outlook 2010 will write the usage count, and periodically apply an algorithm to reduce the usage count to facilitate creating a most frequently used list. However, the MFU list is not implemented, and this attribute is not used.

<9> Section 2.2.3.2.2: Outlook 2007 and Outlook 2010 ignores any digits after the first 3, which means its maximum precision is milliseconds.

<10> Section 2.2.3.2.2: Outlook 2007 and Outlook 2010 will write the last session that this category was applied or changed by the user, but this value is not used.

<11> Section 2.2.3.2.2: Exchange 2003 and Outlook 2003 SP3 do not support renaming categories.

<12> Section 2.2.3.2.2: Outlook 2007 and Outlook 2010 will increment this number on every session. A new session occurs when the user boots Outlook 2007 or Outlook 2010, or when not less than 12 hours has elapsed since the previous session.

<13> Section 2.2.4: view definitions is only supported in Outlook 2003.

<14> Section 2.2.5: Outlook 2003 and Exchange 2003 do not support displaying this description.

<15> Section 2.2.6: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.

<16> Section 2.2.6: Conversation Actions are not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.

<17> Section 2.2.6.1: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.

<18> Section 2.2.6.2: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.

-
- <19> Section 2.2.6.3: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <20> Section 2.2.6.4: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <21> Section 2.2.6.5: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <22> Section 2.2.6.5: Not supported by Outlook 2003 or Outlook 2007.
- <23> Section 2.2.6.5: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010. Outlook 2010 uses a minor version of (140000 + bldnum), where bldnum is a 4-digit build number.
- <24> Section 2.2.6.6: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <25> Section 2.2.6.7: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <26> Section 2.2.6.8: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <27> Section 2.2.6.9: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <28> Section 2.2.6.9: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010. Outlook 2010 sets **PidTagSubject** on the FAI message to a concatenated string, "Conv. Action: " followed by the **PidTagNormalizedSubject** on an E-mail object in the conversation.
- <29> Section 2.2.7: Navigation shortcuts are not supported by Outlook 2003, Exchange 2003, or Exchange 2007.
- <30> Section 2.2.7.16: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010. Outlook 2010 sets the **PidTagWlinkAddressBookEID** property on calendar shortcuts only.
- <31> Section 2.2.7.17: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010. Outlook 2010 sets the **PidTagWlinkAddressBookStoreEID** property on calendar shortcuts only.
- <32> Section 2.2.7.18: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <33> Section 2.2.7.19: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.

-
- <34> Section 3.1.4.1.2: Outlook 2007 and Outlook 2010 use the default values of working from 8 A.M. to 5 P.M., Monday – Friday, in the user’s current system time zone.
- <35> Section 3.1.4.7: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <36> Section 3.1.4.7: Outlook will not attempt to locate matching **E-mail objects** if connected to an online-mode server.
- <37> Section 3.1.4.9.2: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010. Whether Outlook 2010 sets the **PidTagWlinkCalendarColor**, **PidTagWlinkAddressBookEID**, **PidTagWlinkAddressBookStoreEID**, and **PidTagWlinkROGroupType** is dependent upon configuration or administrative settings.
- <38> Section 3.1.5.1: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <39> Section 3.1.5.1.1: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <40> Section xxx: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010. Outlook 2010 maintains a cache of all E-mail objects received in the past 15 minutes.
- <41> Section 3.1.5.2: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <42> Section 3.1.6.1: Outlook has the default duration of 336 hours (14 days).
- <43> Section 3.1.6.1: Outlook implements **conversation action** expiration without timers. Instead, every time an algorithm calls for a **FAI message** to be located, Outlook runs the expiration logic before proceeding with the original algorithm.
- <44> Section 3.2.4.1.2: Exchange 2007 and Exchange 2010 will stop reading the XML document beyond 512 KB.
- <45> Section 3.2.5.1: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <46> Section 3.2.5.2: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.
- <47> Section 3.2.5.3: Not supported by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2010.

Index

Appendix A

Office/Exchange behavior, 107

Introduction, 7

Applicability statement, 14

Glossary, 7

Prerequisites/Preconditions, 13

Protocol overview, 10

References, 9

Relationship to other protocols, 12

Standards assignments, 14

Vendor-extensible fields, 14

Versioning and capability negotiation, 14

Messages, 14

Message syntax, 14

Transport, 14

Protocol details, 73

Client details, 73

Server details, 86

Protocol examples, 91

Configuration data, 91

View definitions, 94

References

Informative references, 10

Normative references, 9

Security, 107

Index of security parameters, 107

Security considerations for implementers, 107