[MS-OXOSFLD]: Special Folders Protocol Specification

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	August 6,	1.01	Updated references to reflect date of initial release.

Corporation	2008		
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	March 4, 2009	1.04	Revised and edited technical content.

Table of Contents

1 Introduction

User data objects are stored by default in a set of common **folders**, referred to as **special folders**.

The Special Folders protocol document specifies:

- The set of special folders shared by client and server implementations of this protocol.
- The specific protocol used to find and interact with each special folder.
- The type of objects stored in each special folder.

1.1 Glossary

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

appointment
complete flag
Calendar folder
Calendar object
contact
Deferred Action Folder (DAF)
delegate
Drafts folder
entry ID
folder
Folder object
folder ID (FID)
free/busy
handle
Inbox folder
journal
Journal object
message ID (MID)
Message object
messaging object
Outbox folder
property
property tag
Receive folder
reminder
remote operation (ROP)
restriction
search criteria
search folder

Sent Items folder
special folder
store
Store object
Task object

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

Boolean
BYTE
DWORD
LONG
ULONG

The following terms are specific to this document:

Common Views folder: The **special folder** that contains the data for default views that are standard for the message **store** and that can be used by any user of a client accessing the message store.

Conflicts folder: The **special folder** that contains **Message object**s that indicate synchronization conflicts between client and server.

Container class: The value of the string property **PidTagContainerClass** on a folder, which indicates the default **Message object** type for the folder**.**

Deleted Items folder: The **special folder** that is the default location for objects that have been deleted.

Finder folder: The **special folder** that contains the default **search folders**.

identification method: The means by which an implementation locates or identifies a particular **special folder**.

Junk E-mail folder: The **special folder** that is the default location for e-mail **Message objects** that are determined to be Junk e-mail by a Junk E-mail Filter.

Local Failures folder: The **special folder** that contains messages that indicate client-side synchronization failures.

Notes folder: The **special folder** that contains Note objects.

Personal Views folder: The **special folder** that contains the data for views defined by a particular user.

Reminders folder: The **special folder** that is a search folder that supports Reminder functionality.

Root folder: The **special folder** that is the top-level folder of the store hierarchy and which contains all other **Folder objects** in that store.

RSS Feeds folder: The **special folder** that contains RSS Feed messages.

Server Failures folder: The **special folder** that contains messages that indicate server-side synchronization failures.

Sync Issues folder: The **special folder** that contains other folders that contain messages that indicate particular issues encountered during synchronization between client and server.

Tasks folder: The **special folder** that contains **Task objects**.

To-Do Search folder: The **special folder** that is used to track **Task objects**.

Top of Personal Folders folder: The **special folder** that is the top folder of the inter-personal message hierarchy, which contains user data folders, including most **special folders**, such as Inbox, and so on.

Tracked Mail Processing folder: The **special folder** that contains objects flagged by the Send and Track feature.

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

1.2 References

1.2.1 Normative References

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

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

[MS-OXCFOLD] Microsoft Corporation, "Folder Object 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-OXCSPAM] Microsoft Corporation, "Spam Confidence Level, Allow and Block Lists Protocol Specification", June 2008.

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

[MS-OXCSYNC] Microsoft Corporation, "Mailbox Synchronization Protocol Specification", June 2008.

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

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

[MS-OXOCFG] Microsoft Corporation, "Configuration Information Protocol Specification", June 2008.

[MS-OXOCNTC] Microsoft Corporation, "Contact Object Protocol Specification", June 2008.

[MS-OXODLGT] Microsoft Corporation, "Delegate Access Configuration Protocol Specification", June 2008.

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

[MS-OXOJRNL] Microsoft Corporation, "Journal Object Protocol Specification", June 2008.

[MS-OXONOTE] Microsoft Corporation, "Note Object Protocol Specification", June 2008.

[MS-OXOPFFB] Microsoft Corporation, "Public Folder–Based Free/Busy Protocol Specification", June 2008.

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

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

[MS-OXOTASK] Microsoft Corporation, "Task-Related Objects Protocol Specification", June 2008.

[MS-OXPHISH] Microsoft Corporation, "Phishing Warning 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.

1.2.2 Informative References

None.

1.3 Protocol Overview

The Special Folders protocol extends the Folder Object protocol [MS-OXCFOLD], which provides users with a general-purpose organizational mechanism based on **Folder objects**. Users have the option of storing particular types of data, such as e-mail messages or Personal Information Manager (PIM) objects such as **appointments**, **contacts**, and so on, in particular **folders** as they see fit. The Special Folders protocol specifies the default set of such folders that an implementation supports out-of–the-box, as well as other non-user-visible **special folders** that support specific folders for certain types of application data, such as **reminders** and views.

Interaction with special folders begins with a determination of whether or not a particular special folder exists within an opened **Store object**, which is specified by the Store Object protocol [MS-OXCSTOR]. This determination is based on the following criteria – note that all **property tags** referenced in this and subsequent sections are specified by [MS-OXPROPS]:

- The appropriate **identification method** has been established for the given special folder.
- The special folder exists in the Store object.
- The value of the **PidTagContainerClass property** of the folder is set to the value defined for that particular special folder.

If these criteria are not met for a particular special folder, an implementation uses the Folder Object protocol to create the folder or, in the case of the **Root folder**, returns an error.

An important aspect of the Special Folders protocol is the method used to identify special folders after they are created. Following these identification methods ensures that the same special folder will continue to be used for a particular type of **messaging object** after the folder is created, and allows an implementation to access special folders in a performant manner. The identification method for each special folder is specified in section 2.2.

The following table lists the set of folders that are special folders, along with the **Container class** for each folder where applicable, and references for further information.

Special folder name	Description	Container class	Related information

Root	The **store** hierarchy's top-level folder, which contains all other Folder objects in that store.	None	[MS-OXCSTOR]
Finder	Contains the default **Search Folders**.	None	[MS-OXOSRCH]
Freebusy Data	Contains the **free/busy** data of the owner.	None	[MS-OXOPFFB]
Top of Personal Folders	The top folder of the interpersonal message hierarchy, which contains user data folders, including most special folders, such as the Inbox, and so on.	None	[MS-OXCSTOR]
Deleted Items	The default location for objects that have been deleted.	"IPF.Note"	[MS-OXOMSG]
Outbox	Outgoing e-mail **Message objects** are placed in this folder at submit time (when the Message object is sent).	"IPF.Note"	[MS-OXOMSG]
Sent Mail	The default location in which copies of e-mail Message objects are placed after they have been submitted (sent).	"IPF.Note"	[MS-OXOMSG]
Inbox	The default location for incoming (received) e-mail Message objects.	"IPF.Note"	[MS-OXOMSG]
Common Views	Contains the data for	None	None

	default views that are standard for the message store and that can be used by any user of a client accessing the message store.		
Personal Views	Contains the data for views defined by a particular user.	None	None
Deferred Action Folder	Contains the deferred action messages that resulted from the execution of client-side rules.	None	[MS-OXORULE]
Calendar	Contains **Calendar objects**, such as appointments.	"IPF.Appointment"	[MS-OXOCAL]
Contacts	Contains Contact objects.	"IPF.Contact"	[MS-OXOCNTC]
Journal	Contains **Journal objects**.	"IPF.Journal"	[MS-OXOJRNL]
Notes	Contains Note objects.	"IPF.StickyNote"	[MS-OXONOTE]
Tasks	Contains **Task objects**.	"IPF.Task"	[MS-OXOTASK]
Reminders	**Search Folder** that supports reminder functionality.	"Outlook.Reminder"	[MS-OXORMDR]
Drafts	The default location for composed e-mail Message objects that have been saved but not sent.	"IPF.Note"	[MS-OXOMSG]

Conflicts	Contains Message objects that indicate synchronization conflicts between client and server.	"IPF.Note"	[MS-OXCSYNC]
Sync Issues	Contains folders that contain messages that indicate particular issues encountered during synchronization between client and server.	"IPF.Note"	[MS-OXCSYNC]
Local Failures	Contains messages that indicate client-side synchronization failures.	"IPF.Note"	[MS-OXCSYNC]
Server Failures	Contains messages that indicate server-side synchronization failures.	"IPF.Note"	[MS-OXCSYNC]
Junk E-mail	Default location for e-mail Message objects determined to be Junk e-mail by a Junk E-mail Filter.	"IPF.Note"	[MS-OXCSPAM]
RSS Feeds	Contains RSS Feed messages.	"IPF.Note.OutlookHomepage"	[MS-OXORSS]
Tracked Mail Processing	Search folder that contains objects flagged by the Send and Track feature.	"IPF.Note"	[MS-OXOFLAG]
To-Do Search	Search folder used for tracking **Task objects**.	"IPF.Task"	[MS-OXOTASK]

1.4 Relationship to Other Protocols

The Special Folders protocol specification relies on an understanding of how to work with **stores**, **folders**, and **properties** (for more details see [MS-OXCSTOR], [MS-OXCFOLD], and [MS-OXCPRPT]), and how these objects are synchronized between the client and server.

1.5 Prerequisites/Preconditions

The Special Folders protocol specification assumes that the messaging client has previously logged on to the messaging server and has acquired a **handle** to the **store** in which the **special folders** are located, as specified in [MS-OXCSTOR].

1.6 Applicability Statement

The Special Folders protocol can be used to locate existing or **store** newly-created well-known object types.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The specific formats for the underlying messages that are sent to and received from the server are specified in [MS-OXCSTOR] for **Store objects**, [MS-OXCFOLD] for **Folder objects**, and [MS-OXCPRPT] for **properties**.

2.2 Message Syntax

The **identification method** for a **special folder** consists of one of the following:

- **Folder IDs (FIDs)** returned from **RopLogon**, as specified in [MS-OXCSTOR]. These FIDs identify the following **folders**:
    - **Root folder**
    - **Finder folder**
    - **Top of Personal Folders folder**

    - **Deleted Items folder**
    - **Outbox folder**
    - **Sent Items folder**
    - **Inbox folder**
    - **Common Views folder**
    - **Personal Views folder**
    - **Deferred Action folder**
- Binary properties that each contain only a single **entry ID**.
- **PidTagAdditionalRenEntryIds**, a MultiBinary **property** in which each indexed value contains an entry ID.
- **PidTagAdditionalRenEntryIdsEx**, a binary property in which the binary data is in its own format, allowing for multiple entry IDs.
- **PidTagFreeBusyEntryIds**, a MultiBinary property in which indexed value 3 contains the entry ID for the Freebusy Data folder. For more details about this property, see [MS-OXOPFFB].
- Use of the Store Object protocol to get or set the identity of the Inbox folder.

Unless otherwise noted, the entry IDs returned by these identification methods MUST be converted to FIDs, as specified in [MS-OXCDATA], before they are used to open a special folder that uses the Folder Object protocol.

2.2.1 Binary Identification Properties

The binary properties that each contain only a single **entry ID** are read or written by using the Property and Stream Object protocol. The following table lists these properties.

Property name	Folder identified
PidTagIpmAppointmentEntryId	Calendar
PidTagIpmContactEntryId	Contacts
PidTagIpmJournalEntryId	Journal
PidTagIpmNoteEntryId	Notes
PidTagIpmTaskEntryId	Tasks
PidTagRemindersOnlineEntryId	Reminders

PidTagIpmDraftsEntryId	Drafts

These properties are read from / written to the **Inbox folder** or **Root folder**. The implementation MUST use the Inbox folder when the **store** is that of the primary messaging user, and it MUST use the Root folder when the store is that of a **delegate** user. These user roles are specified in [MS-OXODLGT].

2.2.2 Format of PidTagAdditionalRenEntryIds

This **MultiBinary property** on the Inbox folder contains the **entry IDs** for several **special folders**. The following table lists the index into the **PidTagAdditionalRenEntryIds** value for each of these special folders.

Index	Folder identified
0x0000	Conflicts
0x0001	Sync Issues
0x0002	Local Failures
0x0003	Server Failures
0x0004	Junk E-mail
0x0005	None. Reserved for use by the Spam Confidence Level, Allow and Block Lists protocol [MS-OXCSPAM] and the Phishing Warning protocol [MS-OXPHISH].

If the implementation encounters an unknown index value in **PidTagAdditionalRenEntryIds**, the implementation MUST ignore and preserve the data in the index entry.

2.2.3 Format of PidTagAdditionalRenEntryIdsEx

Several of the **special folder entry IDs** are identified by this binary **property** on the **Store object** that contains the folders. If present, the value of this property MUST contain an array of blocks that contain the **entry IDs** for these **folders**, in the format specified in the following sections.

2.2.3.1.1 PersistData Block

Name	Type	Size	Description
PersistID	**WORD**	2	Type identifier value for this **PersistData** block. SHOULD be one of **PersistBlockType** values.
DataElementsSize	**WORD**	2	The size in **BYTES** of the **DataElements** field.
DataElements	array of **PersistElement** blocks	varies	0 (zero) or more **PersistElement** blocks.

PersistBlockType values SHOULD be one of those listed in the following table. If a **PersistData** block is encountered where the **PersistID** value is not known to the implementation, the implementation MUST ignore that **PersistData** block and continue processing until either a PERSIST_SENTINEL **PersistID** or the end of the stream is encountered.<1>

Name	Value	Description
RSF_PID_RSS_SUBSCRIPTION	0x8001	Indicates that this block contains data for the **RSS Feeds folder**.
RSF_PID_SEND_AND_TRACK	0x8002	Indicates that this block contains data for the **Tracked Mail Processing folder**.
RSF_PID_TODO_SEARCH	0x8004	Indicates that this block contains data for the **To-Do Search folder**.
PERSIST_SENTINEL	0x0000	Indicates that the implementation MUST stop processing further **PersistData** blocks. PERSIST_SENTINEL is optional.

2.2.3.1.2 PersistElement Block

Name	Type	Size	Description
ElementID	**WORD**	2	Type identifier value for this **PersistElement** block. SHOULD be one of **PersistElementType** values.
ElementDataSize	**WORD**	2	The size in **BYTES** of the **ElementData** field.
ElementData	**BYTE** array of binary data	varies	The data corresponding to this **PersistID\ElementID**.

PersistElementType values SHOULD be one of those listed in the following table. If a **PersistElement** block is encountered where **ElementID** is not known to the implementation, the implementation MUST ignore that **PersistElement** block and continue processing further **PersistElement** blocks until an ELEMENT_SENTINEL **ElementID** or the end of the stream is encountered. The implementation MUST then continue processing additional **PersistData** blocks until either a PERSIST_SENTINEL **PersistID** or the end of the stream is encountered.

Name	Value	Value of ElementDataSize	Description
RSF_ELID_ HEADER	0x0002	0x0004	Indicates that this block's **ElementData** contains a **DWORD** Header value. The interpretation of this value depends on the current block's **PersistID** type. For all **PersistID** types specified in this section, this value MUST be 0 (zero).
RSF_ELID_ENTRYID	0x0001	varies	Indicates that this block contains the **entry ID** of the **folder** indicated by **PersistID**.
ELEMENT_SENTINEL	0x0000	0x0000	Indicates that the implementation MUST stop processing further **PersistElement** blocks for the current **PersistData**

			block.

2.2.4 Inbox Identification

To identify the Inbox, an implementation MUST use **RopGetReceiveFolder** from the Store Object protocol to get the FID for the default **Receive folder** for the **Store object**.

2.2.5 PidTagContainerClass

Several **special folders** use this string **property**, located on the special folder itself, to describe the type of **Message objects** that the **folder** contains. For these folders, an implementation MUST set this property, as shown in the following table.

PropertyValue	Special folder
"IPF.Note"	Deleted Items Outbox Sent Mail Inbox Drafts Conflicts Sync Issues Local Failures Server Failures Junk E-mail Tracked Mail Processing
"IPF.Appointment"	Calendar
"IPF.Contact"	Contacts
"IPF.Journal"	Journal
"IPF.StickyNote"	Notes
"IPF.Task"	Tasks To-Do Search
"Outlook.Reminder"	Reminders
"IPF.Note.OutlookHomepage"	RSS Feeds

3 Protocol Details

Note that the programming elements used in this section, including **restriction** elements such as RES_AND, RELOP_NE, **FID**, **message ID (MID)**, and so on, are specified in [MS-OXCDATA].

3.1 Client and Server Details

Special folders can be opened or created by clients and servers. Except where noted, this section defines constraints to which clients and servers adhere when interacting with special folders.

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.

3.1.1.1 Folder Hierarchy

The following outline depicts a hierarchy for **special folders**.

- Store
    - Root
        - Deferred Action Folder
        - Finder
        - Reminders
        - Tracked Mail Processing
        - To-Do Search
        - Common Views
        - Personal Views
        - Top of Personal Folders
            - Deleted Items
            - Outbox
            - Sent Mail
            - Inbox
            - Calendar

        - Contacts
        - Journal
        - Notes
        - Tasks
        - Drafts
        - Sync Issues
            - Conflicts
            - Local Failures
            - Server Failures
        - Junk E-mail
        - RSS Feeds

3.1.1.2 Search Criteria for Search Special Folders

Some of the **special folders** are **search folders,** as specified in [MS-OXOSRCH], and the functionality supported by these **folders** depends on specific **search criteria**. The detailed specification for each of these is given in section 3.1.4.1, and an abstracted outline of these search criteria is as follows.

For the **Reminders folder**:

- Include all **Message objects** in any folder contained within the **Top of Personal Folders folder**, with the following **restrictions**:
    - The following special folders are excluded from the search:
        - **Deleted Items folder**
        - **Junk Mail folder**
        - **Drafts folder**
        - **Outbox folder**
        - **Conflicts folder**
        - **Local Failures folder**
        - **Server Failures folder**
        - **Sync Issues folder**
    - The value of the **PidTagMessageClass** string **property** does not contain a string with the prefix "IPM.Schedule."

    - The value of the **PidTagMessageFlags LONG** property, defined in [MS-OXCMSG], does not have the MSGFLAG_SUBMIT flag set (that is, submitted Message objects are excluded).
    - The value of the **PidLidReminderSet** property is set to the **Boolean** value 1, OR the value of the **PidLidRecurring** property is set to the **Boolean** value 1.

For the **To-Do Search folder**:

- All Message objects in any folder contained within the Top of Personal Folders folder, with the following **restrictions**:
    - The following special folders are excluded from the search:
        - Deleted Items folder
        - Junk Mail folder
        - Drafts folder
        - Outbox folder
        - Conflicts folder
        - Local Failures folder
        - Server Failures folder
        - Sync Issues folder
    - The message class does not start with "IPM.Appointment" OR "IPM.Activity" OR "IPM.StickyNote".
    - Any one of the following are true:
        - The Message object is a **Task object** [MS-OXOTASK] AND the Task object is owned AND NOT accepted AND the Task object was sent to the currently logged on user.
        - The Message object's Followup Icon index is greater than 0.
        - The Message object's ToDoItem flags include the Active flag.
        - The Message object is an object with the **complete flag** set to True or a completed task.

For the **Tracked Mail Processing folder**:

- All Message objects in any folder contained within the Top of Personal Folders folder, with the following restrictions:
    - The following special folders are excluded from the search:
        - Deleted Items folder
        - Junk Mail folder

        - Drafts folder
        - Outbox folder
        - Conflicts folder
        - Local Failures folder
        - Server Failures folder
        - Sync Issues folder
    - The **PidTagSwappedToDoStore property** exists on the object.
    - The value of the **PidTagMessageFlags** property does not include the MSGFLAG_UNSENT or the MSGFLAG_SUBMIT flags.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

Before an implementation tries to read or write **Message objects** within a **special folder**, the implementation MUST obtain a **handle** to the appropriate special by folder using the following steps:

1. The implementation MUST try to open the special folder by using the appropriate **identification method**, as specified in section 2.2, and the Folder Object protocol, as specified in [MS-OXCFOLD].
2. If the identification method fails, or the special folder does not exist within the **store**, the implementation MUST create the special folder as specified in the following section.

3.1.4.1 Folder Creation

The folder hierarchy specified in section 3.1.1.1 also specifies the dependency chain for **special folder** creation. Before a special folder can be created, all of its container objects in this hierarchy MUST already exist. For example, all special folders (except **Root folder**) depend on the **Store object** and the Root folder. Therefore, the Store object and Root folder MUST exist before any special folder can exist. Also, before any **folder** is created or opened, a HANDLE to the Store object that it contains [MS-OXCSTOR] MUST be obtained by opening the Store object.

To create a special folder, an implementation MUST do the following:

1. Open the parent folder of the desired special folder, as specified in the folder hierarchy in section 3.1.1.1, by using the Folder Object protocol.
2. Create the folder in the opened parent folder, by using the Folder Object protocol and using the special folder name that is appropriate to the implementation's locale, reusing the existing folder if one already exists by that name. The **FID** returned MUST be converted to an **entry ID** as specified by [MS-OXCDATA]. The resulting entry ID of the new folder will be used in the following steps.
3. Establish the **identification method** for the particular special folder created, as specified in section 2.2.
4. If applicable, set the **PidTagContainerClass** string **property** to the appropriate value, as specified in section 2.2.5.
5. Perform any special folder–pecific initialization, as specified in the following sections.

3.1.4.1.1 Creating the Reminders Folder

To complete the creation of the **Reminders folder**, an implementation SHOULD<2> set the **search criteria** for the Reminders folder by using the Search Folder List Configuration protocol [MS-OXOSRCH] and the Folder Object protocol [MS-OXCFLD], such that:

1. The **Top of Personal Folders folder** SHOULD be the only container included in the search.
2. The search applies a **restriction** of type RES_AND with the following two sub-clauses:
    1. A restriction of type RES_AND, with the following sub-clauses – note that a sub-clause is only added if the particular **special folder** already exists within the **store**:
        1. A restriction of type RES_PROPERTY with a relational operator (relop) value of RELOP_NE, comparing the value of the **PidTagParentEntryId property** with the **FID**/**MID** pair of the **Deleted Items folder**.
        2. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Junk Mail folder**.
        3. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Drafts folder**.
        4. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Outbox folder**.

5. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Conflicts folder**.
6. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Local Failures folder**.
7. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Server Failures folder**.
8. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Sync Issues folder**.

2. A restriction of type RES_AND, with the following three sub-clauses:
    1. A restriction of type RES_NOT with the following sub-clause:
        1. A restriction of type RES_AND with the following 2 sub-clauses:
            1. A restriction of type RES_EXIST that specifies the **PidTagMessageClass** property.
            2. A restriction of type RES_CONTENT with a ulFuzzyLevel of FL_PREFIX, comparing the value of PidTagMessageClass property to the string value "IPM.Schedule".
    2. A restriction of type **BitMaskRestriction** with a **BitMapRelOp** value of BMR_EQZ that compares the value of the **PidTagMessageFlags** property to the **ULONG** value mfSubmitted.
    3. A restriction of type RES_OR, with the following two sub-clauses:
        1. A restriction of type RES_PROPERTY, with relop RELOP_EQ, comparing the value of the **PidLidReminderSet** property to the **Boolean** value of 1.
        2. A restriction of type RES_AND, with the following two sub-clauses:
            1. A restriction of type RES_EXIST that specifies the **PidLidRecurring** property.
            2. A restriction of type RES_PROPERTY, with relop RELOP_EQ, comparing the value of the **PidLidRecurring** property to the Boolean value of 1.

3. The search SHOULD run at normal priority relative to other searches, be initiated or restarted if necessary, include child folders, and run without content indexing.

3.1.4.1.2 Creating the To-Do Search Folder

To complete the creation of the **To-Do Search folder**, an implementation MUST set the **search criteria** for the To-Do Search folder by using the Search Folder List Configuration protocol, such that:<3>

1. The **Top of Personal Folders folder** MUST be the only container included in the search.
2. The search applies a restriction of type RES_AND with the following two sub-clauses:
    1. A restriction of type RES_AND, with the following three sub-clauses:
        1. A restriction of type RES_NOT, with the following sub-clause:
            1. A restriction of type RES_CONTENT with a ulFuzzyLevel of FL_PREFIX bitwise or FL_IGNORECASE, comparing the value of the **PidTagMessageClass** property to the string value "IPM.Appointment".
        2. A restriction of type RES_NOT, with the following sub-clause:
            1. A restriction of type RES_CONTENT with a ulFuzzyLevel of FL_PREFIX bitwise or FL_IGNORECASE, comparing the value of the **PidTagMessageClass** property to the string value "IPM.Activity".
        3. A restriction of type RES_NOT, with the following sub-clause:
            1. A restriction of type RES_CONTENT with a ulFuzzyLevel of FL_PREFIX bitwise or FL_IGNORECASE, comparing the value of the **PidTagMessageClass** property to the string value "IPM.StickyNote."
    2. A restriction of type RES_AND, with the following two sub-clauses
        1. A restriction of type RES_AND, with the following sub-clauses – note that a sub-clause is only added if the particular **special folder** already exists within the **store**:
            1. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the **FID**/**MID** pair of the **Deleted Items folder**.
            2. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Junk Mail folder**.

3. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Drafts folder**.
4. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Outbox folder**.
5. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Conflicts folder**.
6. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Local Failures folder**.
7. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Server Failures folder**.
8. restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Sync Issues folder**.

2. A restriction of type RES_OR, with the following four sub-clauses:
    1. A restriction of type RES_AND with the following two sub-clauses:
        1. A restriction of type RES_OR, with the following two sub-clauses:
            1. A restriction of type RES_CONTENT with a ulFuzzyLevel of FL_FULLSTRING bitwise or FL_IGNORECASE, comparing the value of the **PidTagMessageClass** property to the string value "IPM.Task."
            2. A restriction of type RES_CONTENT with a ulFuzzyLevel of FL_PREFIX bitwise or FL_IGNORECASE, comparing the value of the **PidTagMessageClass** property to the string value "IPM.Task."
        2. A restriction of type RES_NOT with the following sub-clause:
            1. A restriction of type RES_AND with the following two sub-clauses:

      1. A restriction of type RES_PROPERTY, with relop RELOP_EQ, comparing the value of the **PidLidTaskState** property to the **LONG** value 2.
      2. A restriction of type RES_PROPERTY, with relop RELOP_NE, comparing the value of the **PidLidTaskAccepted** property to the binary value 1.

2. A restriction of type RES_AND with the following two sub-clauses:
   1. A restriction of type RES_EXIST that specifies the **PidTagFollowupIcon** property.
   2. A restriction of type RES_PROPERTY, with relop RELOP_GT, comparing the value of the **PidTagFollowupIcon** property to the **LONG** value 0 (zero).
3. A restriction of type RES_AND with the following two sub-clauses:
   1. A restriction of type RES_EXIST that specifies the **PidTagToDoItemFlags** property.
   2. A restriction of type BitMaskRestriction with a BitMapRelOp value of BMR_NEZ that compares the value of the **PidTagToDoItemFlags** property to the **ULONG** value 0x00000001.
4. A restriction of type RES_OR, with the following two sub-clauses:
   1. A restriction of type RES_AND with the following three sub-clauses:
      1. A restriction of type RES_OR, with the following two sub-clauses:
         1. A restriction of type RES_NOT, with the following sub-clause:
            1. A restriction of type RES_EXIST that specifies the **PidTagFollowupIcon** property.
         2. A restriction of type RES_PROPERTY, with relop RELOP_EQ, comparing the

value of the **PidTagFollowupIcon** property to the **LONG** value 0 (zero).

2. A restriction of type RES_EXIST that specifies the **PidTagFlagStatus** property.

3. A restriction of type RES_PROPERTY, with relop RELOP_EQ, comparing the value of the **PidTagFlagStatus** property to the **LONG** value 1.

2. A restriction of type RES_AND with the following two sub-clauses:

    1. A restriction of type RES_EXIST that specifies the **PidLidTaskStatus** property.

    2. A restriction of type RES_PROPERTY, with relop RELOP_EQ, comparing the value of the **PidLidTaskStatus** property to the **ULONG** value 2.

3. The search SHOULD run at normal priority relative to other searches, be initiated or restarted if necessary, include child folders, and run without content indexing.

3.1.4.1.3 Creating the Tracked Mail Processing Folder

To complete the creation of the **Tracked Mail Processing folder**, an implementation MUST set the **search criteria** for the Tracked Mail Processing folder by using the Search Folder List Configuration protocol, such that:<4>

1. The **Top of Personal Folders folder** MUST be the only container included in the search.
2. The search applies a restriction of type RES_AND with the following two sub-clauses:
    1. A restriction of type RES_AND, with the following sub-clauses – note that a sub-clause is only added if the particular **special folder** already exists within the **store**:
        1. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId property** with the **FID**/**MID** pair of the **Deleted Items folder**.
        2. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Junk Mail folder**.
        3. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Drafts folder**.

        4. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Outbox folder**.
        5. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Conflicts folder**.
        6. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Local Failures folder**.
        7. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Server Failures folder**.
        8. A restriction of type RES_PROPERTY with a relop value of RELOP_NE, comparing the value of the **PidTagParentEntryId** property with the FID/MID pair of the **Sync Issues folder**.
    2. A restriction of type RES_AND, with the following two sub-clauses:
        1. A restriction of type RES_EXIST that specifies the **PidTagSwappedToDoStore** property.
        2. A restriction of type BitMaskRestriction with a BitMapRelOp value of BMR_EQZ that compares the value of the **PidTagMessageFlags** property to the **ULONG** value including mfUnsent bitwise or mfSubmitted.
3. The search SHOULD run at normal priority relative to other searches, be initiated or restarted if necessary, include child folders, and run without content indexing.

3.1.4.1.4 Creating other Special Folders

If the **special folder** that is being created is one of the following:

- **Deleted Items folder**
- **Junk E-mail folder**
- **Outbox folder**
- **Conflicts folder**
- **Local Failures folder**
- **Server Failures folder**
- **Sync Issues folder**
- **Drafts folder**

An implementation MUST take the following additional steps:

1. Open the **Reminders folder** by using its **identification method** and the Folder Object protocol. If this succeeds, perform the steps in section 3.1.4.1.1.
2. Open the **To-Do Search folder** by using its identification method and the Folder Object protocol. If this succeeds, perform the steps in section 3.1.4.1.2.
3. Open the **Tracked Mail Processing folder** by using its identification method and the Folder Object protocol. If this succeeds, perform the steps in section 3.1.4.1.3.

3.1.5 Message Processing Events and Sequencing Rules

An implementation MUST treat any failure to open the **Root folder** as a failure of the entire Special Folders protocol.

For all other **special folders**, an implementation SHOULD create the special folder if an attempt to open the folder using that **folder's identification method** fails.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

4.1 Opening a Special Folder

Using the **Calendar folder** as an example, opening this **special folder** involves the following:

1. Open the **Inbox folder** by using the Inbox folder **identification method** and the Folder Object protocol. This operation returns a **Folder object**.
2. Get the value of the **PidTagIpmAppointmentEntryId property** from the Inbox folder object returned in step 1 by using the Property and Stream Object protocol. This operation returns an **entry ID** for the Calendar folder.
3. Convert the entry ID returned in step 2 to an **FID** as specified in [MS-OXCDATA], then open this FID by using the Store Object protocol. This operation returns a Folder object for the Calendar folder.

4.1.1 Client Request for Opening a Special Folder

A complete **ROP** request to open the **Calendar folder** would look like the following:

```
0000: 02 01 00 01 01 00 00 00-01 42 0E 44 00
```

RopOpenFolder

ROPid: 0x02

LogonIndex: 0

HandleIndex: 0 (HSOT=0x00000160)

FID: 0001-000001420e44

Note: Open the **Inbox folder**.

OpenModeFlags: 0x00 ReadOnly

```
0000: 07 00 02 00 00 01 00 1D-00 14 00 49 67 03 00 F4
0010: 0F 02 01 72 66 1F 00 E5-36 1F 00 E6 36 1F 00 01
0020: 30 03 00 01 36 03 00 02-36 03 00 03 36 0B 00 0A
0030: 36 1F 00 13 36 02 01 16-36 02 01 D0 36 02 01 D1.
0040: 36 02 01 D2 36 02 01 D3-36 02 01 D4 36 02 01 D5
0050: 36 02 01 D6 36 02 01 D7-36 02 11 D8 36 02 01 D9
0060: 36 03 00 DE 36 02 01 DF-36 02 01 E0 36 03 00 E1
0070: 36 02 11 E4 36 02 01 EB-36 02 01 DA 36
```

RopGetPropertiesSpecific

ROPid: 0x07

LogonIndex: 1

HandleIndex: 1 (HSOT=0x000000e2)

PropertySizeLimit: 0x0000

WantUnicode: 0x0001 (TRUE)

PropCount: 29 (0x1D)

    …

    PidTagIpmAppointmentEntryId

    …

```
0000: 02 01 00 01 01 00 00 00-01 50 4D F6 00
```

RopOpenFolder

ROPid: 0x02

LogonIndex: 1

HandleIndex: 0 (HSOT=0x00000160)

FID: 0001-000001504df6

Note: This is the FID for **Calendar folder** to open

OpenModeFlags: 0x00 ReadOnly

4.1.2 Server Response for Opening a Special Folder

```
0000: 02 01 00 00 00 00 00 00
```

RopOpenFolder

ROPid: 0x02

HandleIndex: 1 (HSOT=0x000000e2)

ReturnValue: ecNone (success) (0x00000000)

HasRulesFlag: 0x00 (FALSE)

IsReplica: 0x00 (FALSE)

Only 256 bytes dumped:

```
0000: 07 01 00 00 00 00 01 00-01 00 00 00 01 42 0E 41
0010: 00 3F 00 00 00 0A 0F 01-04 80 0A 0F 01 04 80 0A
0020: 0F 01 04 80 00 49 00 6E-00 62 00 6F 00 78 00 00
0030: 00 00 01 00 00 00 00 00-00 00 00 00 00 00 00 00
0040: 00 00 00 49 00 50 00 46-00 2E 00 4E 00 6F 00 74
0050: 00 65 00 00 00 0A 0F 01-04 80 00 2E 00 00 00 00
0060: 00 6A 3C B8 FA 3B A9 F0-46 B4 F4 E4 B6 C7 74 45
0070: 09 01 00 02 27 39 56 14-8B EF 4F 98 14 81 7E 2C
0080: 82 BD C2 00 00 01 50 4D-F6 00 00 00 2E 00 00 00
```

```
0090: 00 00 6A 3C B8 FA 3B A9-F0 46 B4 F4 E4 B6 C7 74
00a0: 45 09 01 00 02 27 39 56-14 8B EF 4F 98 14 81 7E
00b0: 2C 82 BD C2 00 00 01 50-4D F7 00 00 00 2E 00 00
00c0: 00 00 00 6A 3C B8 FA 3B-A9 F0 46 B4 F4 E4 B6 C7
00d0: 74 45 09 01 00 02 27 39-56 14 8B EF 4F 98 14 81
00e0: 7E 2C 82 BD C2 00 00 01-50 4D F8 00 00 00 2E 00
00f0: 00 00 00 00 6A 3C B8 FA-3B A9 F0 46 B4 F4 E4 B6
...
```

RopGetPropertiesSpecific

ROPid: 0x07

HandleIndex: 1 (HSOT=0x000000e2)

ReturnValue: ecNone (success) (0x00000000)

…

PropertyArray:

PropCount: 29

…

```
    PidTagIpmAppointmentEntryId          46 Bytes
    0000: 00 00 00 00 6A 3C B8 FA-3B A9 F0 46 B4 F4 E4 B6
    0010: C7 74 45 09 01 00 02 27-39 56 14 8B EF 4F 98 14
    0020: 81 7E 2C 82 BD C2 00 00-01 50 4D F6 00 00
    …
```

Note: The **entry ID** contains 00 00-01 50 4D F6 that is part of the FID: 0001-000001504df6 used for the **RopOpenFolder** in section 4.1.1.

```
0000: 02 01 00 00 00 00 00 00
```

RopOpenFolder

ROPid: 0x02

HandleIndex: 1 (HSOT=0x000000e2)

Note: Successfully opened the **Calendar folder**, and the Calendar folder has a **handle** HSOT=0x000000e2.

HasRulesFlag: 0x00 (FALSE)

IsReplica: 0x00 (FALSE)

…

4.2 Creating a Special Folder

Using the **Calendar folder** as an example, creating this **special folder** involves the following:

1. Open the parent **folder**, in this case the **Top of Personal Folders folder**, as specified in the folder hierarchy in section 3.1.1.1, by using the Top of Personal Folders **identification method** and the Folder Object protocol. This operation returns a **Folder object**.
2. Create a new folder in the opened parent folder, by using the Folder Object protocol, and using the name "Calendar" in English locales, reusing the existing folder if one already exists by that name. The **entry ID** of the newly created folder will be used in the following steps.
3. Open the **Inbox folder** by using the Inbox identification method and the Folder Object protocol. This operation returns a Folder object.
4. Set the value of the **PidTagIpmAppointmentEntryId property** on the Inbox Folder object returned in step 3 to the **entry ID** value of the folder created in step 2, by using the Property and Stream Object protocol.
5. Set the value of the **PidTagContainerClass** string property on the new Calendar folder to the value "IPF.Appointment".

4.2.1 Client Request for Creating a Special Folder

A complete **ROP** request to create the **Calender folder** would look like the following:

```
0000: 02 00 00 01 01 00 00 00-01 42 0E 41 00
```

RopOpenFolder

ROPid: 0x02

LogonIndex: 0

HandleIndex: 0 (HSOT=0x00000059)

FID: 0001-000001420e41

Note: FolderID 4: 0001-000001420e41 **Top of Personal Folders folder** from **RopLogon**.

…

```
0000: 1C 00 00 01 01 01 00 00-43 00 61 00 6C 00 65 00
0010: 6E 00 64 00 61 00 72 00-00 00 43 00 61 00 6C 00
0020: 65 00 6E 00 64 00 61 00-72 00 20 00 43 00 6F 00
0030: 6D 00 6D 00 65 00 6E 00-74 00 00 00
```

RopCreateFolder

ROPid: 0x1C

LogonIndex: 0

HandleIndex: 0 (HSOT=0x00000497)

FolderHandleIndex: 1

FolderType: 0x01

IsUnicodeFolder: 0x01) (TRUE)

OpenIfPreexistingFolder: 0x00 (FALSE)

HasFolderLongTermEID: 0x00 (FALSE)

FolderDisplayName: Calendar

FolderComment: Calendar Comment

```
0000: 02 00 00 01 01 00 00 00-01 42 0E 44 00
```

RopOpenFolder

ROPid: 0x02

LogonIndex: 0

HandleIndex: 0 (HSOT=0x00000059)

FID: 0001-000001420e44

Note: Open the **Inbox folder** with above **FID**.

OpenModeFlags: 0x00 ReadOnly

```
0000: 0A 00 00 36 00 01 00 02-01 D0 36 2E 00 00 00 00
0010: 00 6A 3C B8 FA 3B A9 F0-46 B4 F4 E4 B6 C7 74 45
0020: 09 01 00 02 27 39 56 14-8B EF 4F 98 14 81 7E 2C
0030: 82 BD C2 00 00 01 50 4D-F6 00 00
```

RopSetProperties

ROPid: 0x0A

LogonIndex: 0

HandleIndex: 0 (HSOT=0x000004e4)

Note: HSOT=0x000004e4 is a **handle** to Inbox.

PropertySize: 0x0036 (54)

PropCount: 1 (0x01)

0x36D00102 PidTagIpmAppointmentEntryId 46 Bytes

    0000: 00 00 00 00 6A 3C B8 FA-3B A9 F0 46 B4 F4 E4 B6
    0010: C7 74 45 09 01 00 02 27-39 56 14 8B EF 4F 98 14
    0020: 81 7E 2C 82 BD C2 00 00-01 50 4D F6 00 00

```
0000: 0A 00 00 26 00 01 00 1F-00 13 36 49 00 50 00 46
0010: 00 2E 00 41 00 70 00 70-00 6F 00 69 00 6E 00 74
0020: 00 6D 00 65 00 6E 00 74-00 00 00
```

RopSetProperties

ROPid: 0x0A

LogonIndex: 0

HandleIndex: 0 (HSOT=0x0000042e)

PropertySize: 0x0026 (38)

PropCount: 1 (0x01)

PidTagContainerClass

IPF.Appointment

4.2.2 Server Response for Creating a Special Folder

```
0000: 02 01 00 00 00 00 00 00
```

RopOpenFolder

ROPid: 0x02

HandleIndex: 1 (HSOT=0x00000497)

Note: Handle HSOT=0x00000497 of **Top of Personal Folders folder** is used to create the **Calendar folder** in section 4.2.1, Client Request **RopCreateFolder**.

ReturnValue: ecNone (success) (0x00000000)

HasRulesFlag: 0x00 (FALSE)

IsReplica: 0x00 (FALSE)

```
0000: 1C 01 00 00 00 00 01 00-00 00 01 50 4D F6 00
```

RopCreateFolder

ROPid: 0x1C

HandleIndex: 1 (HSOT=0x0000042e)

Note: HSOT=0x0000042e is the handle to the Calendar folder that was created.

ReturnValue: ecNone (success) (0x00000000)

FID: 0001-000001504df6

IsExistingFolder: 0x00 (FALSE)

```
0000: 02 01 00 00 00 00 00 00
```

RopOpenFolder

ROPid: 0x02

HandleIndex: 1 (HSOT=0x000004e4)

Note: HSOT=0x000004e4 is the handle of the **Inbox folder**. It is used in 4.2.1, Client Request **RopSetProperties**.**PidTagIpmAppointmentEntryId** of Calendar folder.

ReturnValue: ecNone (success) (0x00000000)

HasRulesFlag: 0x00 (FALSE)

IsReplica: 0x00 (FALSE)

```
0000: 0A 00 00 00 00 00 00 00
```

RopSetProperties

ROPid: 0x0A

HandleIndex: 0 (HSOT=0x0000042e)

ReturnValue: ecNone (success) (0x00000000)

ProblemPropertyTagCount: 0

ProblemPropertyTagArray:

```
0000: 0A 00 00 00 00 00 00 00
```

RopSetProperties

ROPid: 0x0A

HandleIndex: 0 (HSOT=0x0000042e)

ReturnValue: ecNone (success) (0x00000000)

ProblemPropertyTagCount: 0

ProblemPropertyTagArray:

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the Special Folders protocol. General security considerations pertaining to the underlying protocols apply (see [MS-OXCSTOR], [MS-OXCFOLD], and [MS-OXCPRPT]).

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 2003
- Microsoft Exchange Server 2003
- Microsoft Office 2007
- Microsoft Exchange Server 2007

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 2.2.3.1.1: Microsoft Office Outlook 2003 clients and Exchange 2003 servers do not recognize as special the RSS Feeds, Tracked Mail Processing, and To-Do Search folders.

<2> Section 3.1.4.1.1: Outlook 2003 sets the search criteria to include only the Calendar, Tasks, Inbox, and Contacts folders, and it only sets a restriction for **PidLidReminderSet** and **PidLidRecurring**, and it does not include RECURSIVE_SEARCH.

<3> Section 3.1.4.1.2: Outlook 2003 clients and Exchange 2003 servers do not recognize the To-Do search criteria.

<4> Section 3.1.4.1.3: Outlook 2003 clients and Exchange 2003 servers do not recognize the Tracked Mail search criteria.

Index