[MS-OXOCAL]:
Appointment and Meeting Object Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1	Major	Initial Availability.
04/25/2008	0.2	Minor	Revised and updated property names and other technical content.
06/27/2008	1.0	Major	Initial Release.
08/06/2008	1.0.1	Editorial	Revised and edited technical content.
09/03/2008	1.0.2	Editorial	Updated references.
12/03/2008	1.0.3	Editorial	Revised and edited technical content.
12/02/2009	1.0.4	Editorial	Revised and edited technical content.
04/10/2009	2.0	Major	Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	3.1.0	Minor	Updated the technical content.

Table of Contents

1 Introduction

The concept of calendaring involves enabling users to manage their schedules electronically. Users can create **events** on their calendars and optionally request others to attend. The events can be made to recur at specific intervals. Upon receiving an invitation to a **calendar** event, users can accept, decline, or propose a different date and/or time for the event. Delegation enables one user to manage the calendar of another user.

The Appointment and Meeting Object protocol specifies how to extend the [MS-OXCMSG] protocol for use with calendaring. This document also specifies the following:

- The format for storing events as **Calendar objects**.
- A process for retrieval of those objects by a client or server.
- A process for scheduling other users.
- A process for allowing another user to manage the calendar.
- A process for scheduling commonly shared **resources**.

1.1 Glossary

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

Address Book object
appointment
Appointment object
Attachment object
attendee
Bcc recipient
binary large object (BLOB)
Calendar folder
Calendar object
Coordinated Universal Time (UTC)
counter proposal
delegate
Delegate Information object
delegator
EntryID
exception
Embedded Message object
Exception Attachment object
Exception Embedded Message object
Exception object
GUID
handle
informational update
instance
little-endian
meeting
Meeting object
Meeting Cancellation object
meeting-related object
meeting request

Meeting Request object
Meeting Response object
Meeting Update object
Meeting Workspace
Message object
Out of Office (OOF)
organizer
property
public folder
Recurring Calendar object
recurrence pattern
recurring series
resource
Rich Text Format (RTF)
Sent Mail folder
signal time
single instance
special folder
store
Task object
Unicode

The following terms are specific to this document:

Calendar special folder: A Calendar folder in a user's mailbox that meetings will be created in by default. For details about special folders, see [MS-OXOSFLD].

floating appointment: An appointment that starts and ends at the same local time regardless of any time zone considerations.

full update: A Meeting Update object that includes a change to the date and/or time, or recurrence pattern, and which requires a response from attendees.

Meeting Cancellation object: A Message object that is sent to attendees when the organizer of a meeting cancels a previously scheduled event.

Meeting Forward Notification object: A Message object sent to an organizer when an attendee forwards a meeting request.

meeting update: An instance of a Meeting Update object.

optional attendee: An attendee of an event whom the organizer lists as an optional participant.

orphan instance: An instance of a recurring series that is in a Calendar folder without the recurring series. For all practical purposes, this is a single instance.

required attendee: An attendee of an event whom the organizer lists as a mandatory participant.

sendable attendee: An attendee to whom a meeting request or meeting update will be sent. A sendableattendee can be a required or optionalattendee, or a resource.

sequence number: The revision number of a Meeting object. The is used to determine the most recent meeting update that was sent by the organizer.

series: See recurring series.

significant change: A change made by an organizer to a Meeting object that requires a Meeting Update object to be sent.

time zone update: Any change to a time zone that occurs when a time zone changes the dates in which it observes Daylight Saving Time (DST) or changes its offset from UTC.

unsendable attendee: An attendee to whom meeting-related objects will not be sent.

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

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

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

[MS-MEETS] Microsoft Corporation, "Meetings Web Services Protocol Specification", April 2008, http://msdn.microsoft.com/en-us/library/cc313057.aspx.

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

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

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

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

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

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

[MS-OXPROPS] Microsoft Corporation, "Exchange Server Protocols Master Property List", 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 Appointment and Meeting object protocol specifies the following:

- The **Message objects** that are required for working with a user's electronic schedule, as reflected in the contents of a **calendar folder**.
- How scheduled events are communicated among users, including the **organizer** and **attendees**.
- The interaction between a **delegate** and the **delegator's** calendar.

1.3.1 Protocol Objects

The Message objects that are specified by the Appointment and Meeting object protocol can be classified as one of the following two types of objects:

- Calendar objects, which are objects that are created and reside in a Calendar folder. The two Calendar object types are **Appointment objects** and **Meeting objects**.
- **meeting-related objects**, which are objects that relay Meeting object information from organizer to attendees and vice versa. These include **Meeting Request objects**, **Meeting Update objects**, **Meeting Cancellation objects**, **Meeting Response objects**, and **Meeting Forward Notification objects**.

1.3.1.1 Appointment Object

The Appointment object contains details of an event, such as a description, notes, date and time, **reminder** date and time, status, and more. The event that is specified by the Appointment object can be a **single instance** or a recurring event with or without **exceptions**.

1.3.1.1.1 Exceptions

An exception represents a modified **instance** of a recurring event. This could be as simple as extra data in the body, or it could be more complicated, such as a change in date/time or location. An exception is defined by an **Exception Attachment object** and an **Exception Embedded Message object**.

1.3.1.2 Meeting Object

A Meeting object extends the Appointment object to contain attendees in addition to the organizer. The Meeting object is created, owned, and managed by an organizer.

1.3.1.2.1 Attendees

Attendees are people or resources that are invited by the organizer to an event. Attendees can be of three types: required, optional, and resource. Attendees, of any type, can be further categorized as sendable or unsendable. **Meeting requests** are sent to **sendable attendees** but not to **unsendable attendees**.

1.3.1.3 Meeting Request Object

The organizer invites one or more users to attend a **meeting** by sending a Meeting Request object. This object is sent to each sendable attendee to communicate the event details.

1.3.1.4 Meeting Response Object

When an attendee receives a meeting request, he or she can accept, tentatively accept, or decline the invitation. The attendee sends a Meeting Response object back to the organizer that indicates their response choice. With the response, the attendee can propose a new date and/or time that works better for the attendee.

1.3.1.5 Meeting Update Object

If the organizer decides to make changes to a previously scheduled meeting, the organizer sends a special type of Meeting Request object, referred to as the Meeting Update object, to communicate these changes. If a change occurs to the date and/or time or **recurrence pattern**, it is considered a **full update** and attendees are required to re-respond. Other changes, such as additional agenda details, are considered **informational updates** and do not require a new response.

1.3.1.6 Meeting Cancellation Object

The organizer sends a Meeting Cancellation object to notify attendees that a previously scheduled event will not take place.

1.3.1.7 Meeting Forward Notification Object

When an attendee forwards a Meeting Request object to new attendees, the organizer is notified of the new attendees through a Meeting Forward Notification object.

1.4 Relationship to Other Protocols

The Appointment and Meeting object protocol extends the [MS-OXCMSG] protocol for use with Calendar objects and relies on [MS-OXOMSG] for **message** transport and delivery.

1.5 Prerequisites/Preconditions

The Appointment and Meeting Object protocol assumes that the client has previously acquired a **handle** to the object on which it intends to operate. It also assumes that the client has acquired a handle to the Calendar folder to access Calendar objects when required. It relies on an understanding of how to work with **folders**, messages, **recipients**, and **tables**. For more details, see [MS-OXCPRPT], [MS-OXCMSG], and [MS-OXCFOLD].

1.6 Applicability Statement

The Appointment and Meeting object protocol is appropriate for clients and servers that manage user **appointments** and meetings and their associated resources.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

This protocol does not provides any vendor extensibility beyond what is already specified in [MS-OXCMSG].

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Appointment and Meeting object protocol uses the protocols specified in [MS-OXCPRPT] and [MS-OXCMSG] as its primary transport mechanism.

2.2 Message Syntax

Calendar objects and meeting-related objects can be created and modified by clients and servers. This section defines the constraints under which both clients and servers operate.

Clients operate on Calendar objects and meeting-related objects by using the Message and Attachment Object protocol, as specified in [MS-OXCMSG]. How servers operate on these objects is implementation-dependent, but the results of any such operations MUST be exposed to clients as specified by the Appointment and Meeting object protocol.

Unless otherwise specified, Calendar objects and meeting-related objects MUST adhere to all **property** constraints specified in [MS-OXPROPS] and all property constraints specified in [MS-OXCMSG]. An object can contain other properties, as specified in [MS-OXPROPS], but these properties do not have any impact on the appointment and Meeting object protocol <1><2><3>

When a property is referred to as "read-only for the client", it means that a client SHOULD NOT attempt to change the value of this property and a server returns an error and ignores any request to change the value of this property.

2.2.1 Common Properties

Unless otherwise noted, the objects specified in the Appointment and Meeting object protocol include the common properties, as specified in [MS-OXCPRPT]. The objects also include the common properties, as specified in [MS-OXCMSG]. The objects SHOULD also set the common properties, as specified in [MS-OXOMSG].

This section describes the properties that are common to all object types in the appointment and Meeting object protocol. Unless otherwise specified, the properties listed in this section exist on all Calendar objects and meeting-related objects.

2.2.1.1 PidLidAppointmentSequence

Type: **PtypInteger32**, unsigned

Specifies the **sequence number** of a Meeting object. A Meeting object begins with the sequence number set to 0 (zero) and is incremented each time the organizer sends out a Meeting Update object. The sequence number is copied onto the Meeting Response object so that the client or server knows which version of the meeting is being responded to. For more details about when and how a client increments the sequence number, see section 3.1.5.4.

2.2.1.2 PidLidBusyStatus

Type: **PtypInteger32**

Specifies the availability of a user for the event described by the object and MUST be one of the values specified in the following table.

Status	Value	Description
olFree	0x00000000	The user is available.
olTentative	0x00000001	The user has a **Tentative** event scheduled.
olBusy	0x00000002	The user is **Busy**.
olOutOfOffice	0x00000003	The user is **Out of Office (OOF)**.

2.2.1.3 PidLidAppointmentAuxiliaryFlags

Type: **PtypInteger32**

Specifies a bit field that describes the auxiliary **state** of the object. This property is not required. The following are the individual **flags** that can be set.

C (auxApptFlagCopied, 0x00000001): This flag indicates that the Calendar object was copied from another Calendar folder. <4>

R (auxApptFlagForceMtgResponse, 0x00000002): This flag on a Meeting Request object indicates that the client or server can require that a Meeting Response object be sent to the organizer when a response is chosen.

F (auxApptFlagForwarded, 0x00000004): This flag on a Meeting Request object indicates that it was forwarded by the organizer or another recipient, rather than sent directly from the organizer.

2.2.1.4 PidLidLocation

Type: **PtypString**

Specifies the location of the event. This property is not required.

2.2.1.5 PidLidAppointmentStartWhole

Type: **PtypTime**

Specifies the start date and time of the event in **UTC** and MUST be less than the value of the PidLidAppointmentEndWhole property. For a **recurring series**, this property is the start date and time of the first instance according to the recurrence pattern. Note that for some appointments, the value of this time property is not interpreted strictly as a UTC time. See section 3.1.5.5 for more information.

2.2.1.6 PidLidAppointmentEndWhole

Type: **PtypTime**

Specifies the end date and time for the event in UTC and MUST be greater than the value of the PidLidAppointmentStartWhole property. For a recurring series, this property is the end date and time of the first instance according to the recurrence pattern. Note that for some appointments, the value of this time property is not interpreted strictly as a UTC time. See section 3.1.5.5 for more information.

2.2.1.7 PidLidAppointmentDuration

Type: **PtypInteger32**

Specifies the length of the event, in minutes. This property is not required. If set, the value MUST be the number of minutes between the value of the PidLidAppointmentStartWhole and PidLidAppointmentEndWhole properties<5>.

2.2.1.8 PidLidAppointmentColor

Type: **PtypInteger32**

Specifies the color to be used when displaying the Calendar object. A client or server SHOULD set this value for backward compatibility with older clients; however, it can instead display the Calendar object based on the value of the PidNameKeywords property, as specified in [MS-OXCMSG]. When set, this property MUST have one of the values specified in the following table.

Value	Color
0x00000000	None
0x00000001	Red
0x00000002	Blue
0x00000003	Green
0x00000004	Grey
0x00000005	Orange
0x00000006	Cyan
0x00000007	Olive
0x00000008	Purple
0x00000009	Teal
0x0000000A	Yellow

2.2.1.9 PidLidAppointmentSubType

Type: **PtypBoolean**

Specifies whether the event is an all-day event, as specified by the user. A value of TRUE indicates that the event is an all-day event, in which case the start time and end time MUST be midnight so that the duration is a multiple of 24 hours and is at least 24 hours. A value of FALSE or the absence of this property indicates that the event is not an all-day event. The client or server can not infer the value as TRUE when a user happens to create an event that is 24 hours long, even if the event starts and ends at midnight.

2.2.1.10 PidLidAppointmentStateFlags

Type: **PtypInteger32**

Specifies a bit field that describes the state of the object. This property is not required. The following are the individual flags that can be set.

M (asfMeeting, 0x00000001): This flag indicates that the object is a Meeting object or a meeting-related object.

R (asfReceived, 0x00000002): This flag indicates that the represented object was received from someone else.

C (asfCanceled, 0x00000004): This flag indicates that the Meeting object that is represented by the object has been canceled.

2.2.1.11 PidLidResponseStatus

Type: **PtypInteger32**

Specifies the response status of an attendee, and MUST be one of the values listed in the following table.

Response status	Value	Description
respNone	0x00000000	No response is required for this object. This is the case for Appointment objects and Meeting Response objects.
respOrganized	0x00000001	This Meeting object belongs to the organizer.
respTentative	0x00000002	This value on the attendee's Meeting object indicates that the attendee has tentatively accepted the Meeting Request object.
respAccepted	0x00000003	This value on the attendee's Meeting object indicates that the attendee has accepted the Meeting Request object.
respDeclined	0x00000004	This value on the attendee's Meeting object indicates that the attendee has declined the Meeting Request object.
respNotResponded	0x00000005	This value on the attendee's Meeting object indicates that the attendee has not yet responded. This value is on the Meeting Request object, Meeting Update object, and Meeting Cancellation object.

2.2.1.12 PidLidRecurring

Type: **PtypBoolean**

Specifies whether the object represents a recurring series. A value of TRUE indicates that the object represents a recurring series. A value of FALSE, or the absence of this property, indicates that the object represents either a single instance or an exception (including an **orphan instance**). Note the difference between this property and the property PidLidIsRecurring.

2.2.1.13 PidLidIsRecurring

Type: **PtypBoolean**

Specifies whether the object is associated with a recurring series. A value of TRUE indicates that the object represents either a recurring series or an exception (including an orphan instance). A value of FALSE, or the absence of this property<6>, indicates that the object represents a single instance. Note the difference between this property and the property PidLidRecurring.

2.2.1.14 PidLidClipStart

Type: **PtypTime**

For single instance Calendar objects, this property specifies the start date and time of the event in UTC. For a recurring series, this property specifies midnight on the date of the first instance, in UTC.

2.2.1.15 PidLidClipEnd

Type: **PtypTime**

For single instance Calendar objects, the property specifies the end date and time of the event in UTC. For a recurring series, this property specifies midnight on the date of the last instance of the recurring series in UTC, unless the recurring series has no end, in which case the value MUST be 31 August 4500, 11:59 P.M.

2.2.1.16 PidLidAllAttendeesString

Type: **PtypString**

Specifies a list of all the attendees except for the organizer, including resources and unsendable attendees. The value for each attendee is the PidTagDisplayName property of the attendee's Address Book object. Separate entries are delimited by a semicolon followed by a space. This property is not required.

2.2.1.17 PidLidToAttendeesString

Type: **PtypString**

This property contains a list of all the sendable attendees who are also **required attendees**. The value for each attendee is the PidTagDisplayName property of the attendee's **Address Book object**. Separate entries are delimited by a semicolon followed by a space. This property is not required.

2.2.1.18 PidLidCcAttendeesString

Type: **PtypString**

This property contains a list of all the sendable attendees who are also **optional attendees**. The value for each attendee is the PidTagDisplayName property of the attendee's Address Book object. Separate entries are delimited by a semicolon followed by a space. This property is not required.

2.2.1.19 PidLidNonSendableTo

Type: **PtypString**

This property contains a list of all the unsendable attendees who are also required attendees. The value for each attendee is the PidTagDisplayName property of the attendee's Address Book object. Separate entries are delimited by a semicolon followed by a space. This property is not required.

2.2.1.20 PidLidNonSendableCc

Type: **PtypString**

This property contains a list of all the unsendable attendees who are also optional attendees. The value for each attendee is the PidTagDisplayName property of the attendee's Address Book object. Separate entries are delimited by a semicolon followed by a space. This property is not required.

2.2.1.21 PidLidNonSendableBcc

Type: **PtypString**

This property contains a list of all the unsendable attendees who are also resources. The value for each attendee is the PidTagDisplayName property of the attendee's Address Book object. Separate entries are delimited by a semicolon followed by a space. This property is not required.

2.2.1.22 PidLidNonSendToTrackStatus

Type: **PtypMultipleInteger32**

This property contains the value from the response table (see section 2.2.1.11) for each attendee listed in the PidLidNonSendableTo property. This property is required only when the PidLidNonSendableTo property is set. The number of values in this property MUST equal the number of values in the PidLidNonSendableTo property. Each **PtypInteger32** value in this property corresponds to the attendee in the PidLidNonSendableTo property at the same index. This property is not required.

2.2.1.23 PidLidNonSendCcTrackStatus

Type: **PtypMultipleInteger32**

This property contains the value from the response table (see section 2.2.1.11) for each attendee listed in the PidLidNonSendableCc property. This property is required only when the PidLidNonSendableCc property is set. The number of values in this property MUST equal the number of values in the PidLidNonSendableCc property. Each **PtypInteger32** value in this property corresponds to the attendee in the PidLidNonSendableCc property at the same index. This property is not required.

2.2.1.24 PidLidNonSendBccTrackStatus

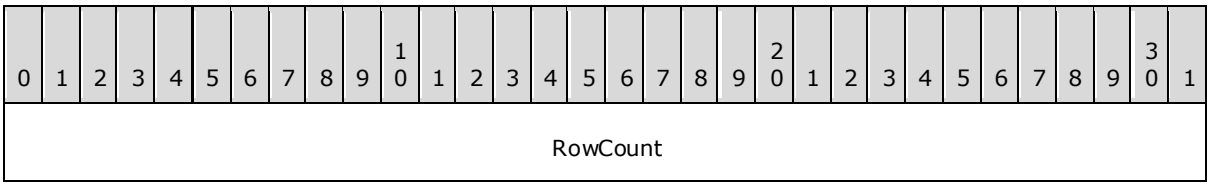
Type: **PtypMultipleInteger32**

This property contains the value from the response table (see section 2.2.1.11) for each attendee listed in the PidLidNonSendableBcc property. This property is required only when the PidLidNonSendableBcc property is set. The number of values in this property MUST equal the number of values in the PidLidNonSendableBcc property. Each **PtypInteger32** value in this property corresponds to the attendee in the PidLidNonSendableBcc property at the same index. This property is not required.

2.2.1.25 PidLidAppointmentUnsendableRecipients

Type: **PtypBinary**

This property contains a list of unsendable attendees. This property is not required, but SHOULD be set. <7> It has the following format:



RecipientRow[1-..RowCount]

RowCount (4 bytes): The count of RecipientRow.

RecipientRow[1-..RowCount] (4 bytes): A list recipient of table **rows**. For details, see [MS-OXCMSG]. See also the additional properties in section 2.2.3.9 that can be set on **RecipientRows** for Calendar objects and meeting-related objects.

2.2.1.26 PidLidAppointmentNotAllowPropose

Type: **PtypBoolean**

A value of TRUE for this property indicates that attendees are not allowed to propose a new date and/or time for the meeting. A value of FALSE or the absence of this property indicates that the attendees are allowed to propose a new date and/or time. This property is only meaningful on Meeting objects, Meeting Request objects, and Meeting Update objects.

2.2.1.27 PidLidGlobalObjectId

Type: **PtypBinary**

Specifies the unique **identifier** of the Calendar object. After it is set for a Calendar object, the value of this property MUST NOT change. The fields in this **BLOB** are specified in the following table. All fields have **little-endian** byte order.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Byte Array ID																															
...																															
...																															
...																															
YH								YL								M								D							
Creation Time																															
...																															
X																															
...																															

Size
Data (variable)
...

Byte Array ID (16 bytes): An array of 16 bytes identifying this BLOB as a Global Object ID. The byte array MUST be as follows: 0x04, 0x00, 0x00, 0x00, 0x82, 0x00, 0xE0, 0x00, 0x74, 0xC5, 0xB7, 0x10, 0x1A, 0x82, 0xE0, 0x08.

YH (1 byte): The high-ordered byte of the 2-byte Year from the PidLidExceptionReplaceTime property if the object represents an exception; otherwise, zero.

YL (1 byte): The low-ordered byte of the 2-byte Year from the PidLidExceptionReplaceTime property if the object represents an exception; otherwise, zero.

M (1 byte): The Month from the PidLidExceptionReplaceTime property if the object represents an exception; otherwise, zero. If it represents an exception, the value MUST be one of those listed in the following table.

Value	Month
0x01	January
0x02	February
0x03	March
0x04	April
0x05	May
0x06	June
0x07	July
0x08	August
0x09	September
0x0A	October
0x0B	November
0x0C	December

D (1 byte): The Day of the month from the PidLidExceptionReplaceTime property if the object represents an exception; otherwise, zero.

Creation Time (8 bytes): The date and time that this Global Object ID was generated, as a [MS-DTYP]: **FILETIME**. This component can be all zeros.

X (8 bytes): Reserved, MUST be all zeroes.

Size (4 bytes): A **LONG** value that defines the size of the Data component.

Data (variable): An array of bytes that ensures the uniqueness of the Global Object ID among all Calendar objects in all **mailboxes**.

2.2.1.28 PidLidCleanGlobalObjectId

Type: **PtypBinary**

The format of this property is the same as that of PidLidGlobalObjectId. The value of this property MUST be equal to the value of PidLidGlobalObjectId, except the *YH*, *YL*, *M*, and *D* fields MUST all be zero. All objects that refer to an instance of a recurring series (including an orphan instance), as well as the recurring series itself, will have the same value for this property.

2.2.1.29 PidTagOwnerAppointmentId

Type: **PtypInteger32**

Specifies a quasi-unique value among all Calendar objects in a user's Mailbox. The value of this property can assist a client or server in finding a Calendar object, but is not guaranteed to be unique among all objects.<8> This property is not required on objects.

2.2.1.30 PidTagStartDate

Type: **PtypTime**

For backward compatibility with older clients, this property SHOULD be set, and when set, it MUST be equal to the value of the PidLidAppointmentStartWhole property.

2.2.1.31 PidTagEndDate

Type: **PtypTime**

For backward compatibility with older clients, this property SHOULD be set, and when set, it MUST be equal to the value of the PidLidAppointmentEndWhole property.

2.2.1.32 PidLidCommonStart

Type: **PtypTime**

The value of this property MUST be equal to the value of the PidLidAppointmentStartWhole property.

2.2.1.33 PidLidCommonEnd

Type: **PtypTime**

The value of this property MUST be equal to the value of the PidLidAppointmentEndWhole property.

2.2.1.34 PidLidOwnerCriticalChange

Type: **PtypTime**

Specifies the date and time at which a Meeting Request object was sent by the organizer. The value is specified in UTC.

2.2.1.35 PidLidIsException

Type: **PtypBoolean**

A value of TRUE for this property indicates that the object represents an exception (including an orphan instance). A value of FALSE indicates that the object represents a recurring series or a single instance. The absence of this property for any object indicates a value of FALSE except for the Exception Embedded Message object, which assumes a value of TRUE.

2.2.1.36 PidTagResponseRequested

Type: **PtypBoolean**

When the value of this property is FALSE, Meeting Response objects are not sent to the organizer. When the value of this property is TRUE, and the client or server automatically responds (see section 2.2.11.2, 2.2.11.3, and 2.2.11.4), a Meeting Response object is sent to the organizer. Otherwise, when the value is TRUE, the client or server can <9> send a Meeting Response object.

2.2.1.37 PidTagReplyRequested

Type: **PtypBoolean**

This property MUST have the same value as PidTagResponseRequested for Calendar objects.

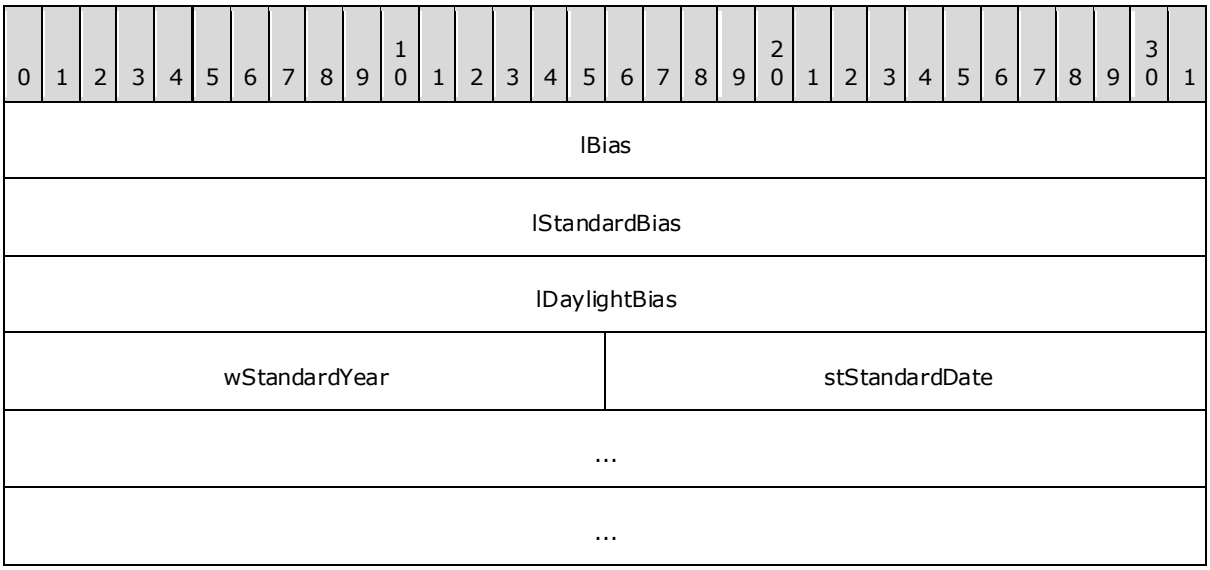
2.2.1.38 Best Body Properties

These properties contain the contents of the Calendar objects or meeting-related objects. The contents SHOULD use the **RTF** properties [MS-OXRTFCP] for objects that are specified by the appointment and Meeting object protocol. When stored and retrieved, **best body** guidance, as specified in [MS-OXBBODY], is to be followed.

2.2.1.39 PidLidTimeZoneStruct

Type: **PtypBinary**

This property is set on a recurring series to specify time zone information. This property specifies how to convert time fields between local time and UTC. The fields in this BLOB are encoded in little-endian byte order.



...	
...	wDaylightYear
stDaylightDate	
...	
...	
...	

lBias (4 bytes): The time zone's offset in minutes from UTC.

lStandardBias (4 bytes): The offset in minutes from *lBias* during standard time.

lDaylightBias (4 bytes): The offset in minutes from *lBias* during daylight saving time.

wStandardYear (2 bytes): This field matches the *stStandardDate's* wYear member.

stStandardDate (16 bytes): *SYSTEMTIME* structure as specified in [MS-DTYP]. This field contains the date and local time that indicate when to begin using the *lStandardBias*.

If the time zone does not support daylight saving time, the *wMonth* member in the **SYSTEMTIME** structure MUST be 0 (zero). If the *wYear* member is not 0 (zero), the date is interpreted as an absolute date that only occurs once. If the *wYear* member is 0 (zero), the date is interpreted as a relative date that occurs yearly. The *wHour* and *wMinute* members are set to the transition time; the *wDayOfWeek* member is set to the appropriate weekday, and the *wDay* member is set to indicate the occurrence of the day of the week within the month (1 to 5, where 5 indicates the final occurrence during the month if that day of the week does not occur 5 times).

wDaylightYear (2 bytes): This field matches the *stDaylightDate'swYear* field.

stDaylightDate (16 bytes): **SYSTEMTIME** structure as specified in [MS-DTYP]. This field contains the date and local time that indicate when to begin using the *lDaylightBias*. This field has the same format and constraints as the *stStandardDate* field.

2.2.1.40 PidLidTimeZoneDescription

Type: **PtypString**

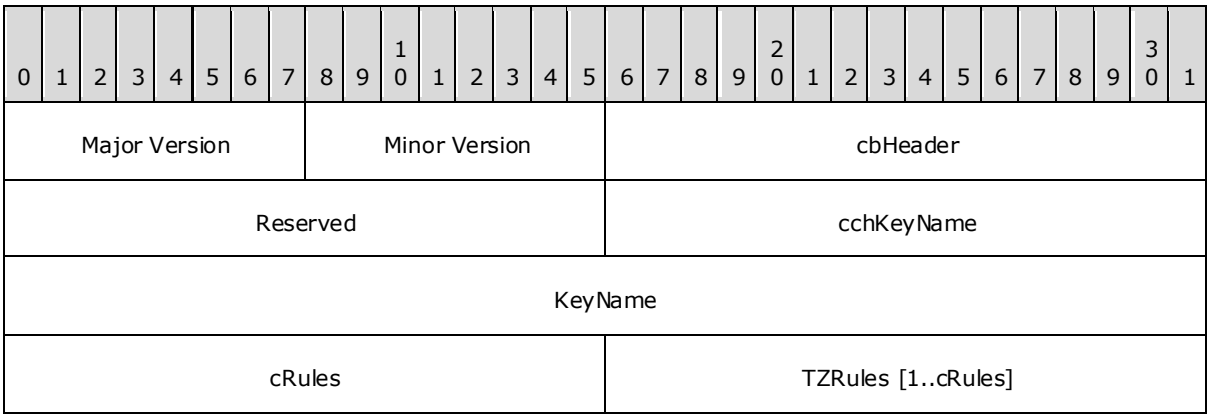
Specifies a human-readable description of the time zone that is represented by the data in the PidLidTimeZoneStruct property.

2.2.1.41 PidLidAppointmentTimeZoneDefinitionRecur

Type: **PtypBinary**

Specifies time zone information that describes how to convert the meeting date and time on a recurring series to and from UTC. If this property is set, but it has data that is inconsistent with the

data that is represented by PidLidTimeZoneStruct, then the client uses PidLidTimeZoneStruct instead of this property.<10><11>



Major Version (1 byte): This field is set to 0x02.

Minor Version (1 byte): This field is set to 0x01.

cbHeader (2 bytes): The count of bytes contained in *Reserved*, *cchKeyName*, *KeyName*, and *cRules*.

Reserved (2 bytes): This Word field MUST be set to 0x0002.

cchKeyName (2 bytes): This Word field represents the count of characters in the *KeyName* field that follows.

KeyName (4 bytes): **Unicode** string that identifies the associated time zone. The string is not localized but instead is set to the unique name of the desired time zone <12>. This string has a maximum length of 260 characters, and it is not null terminated.

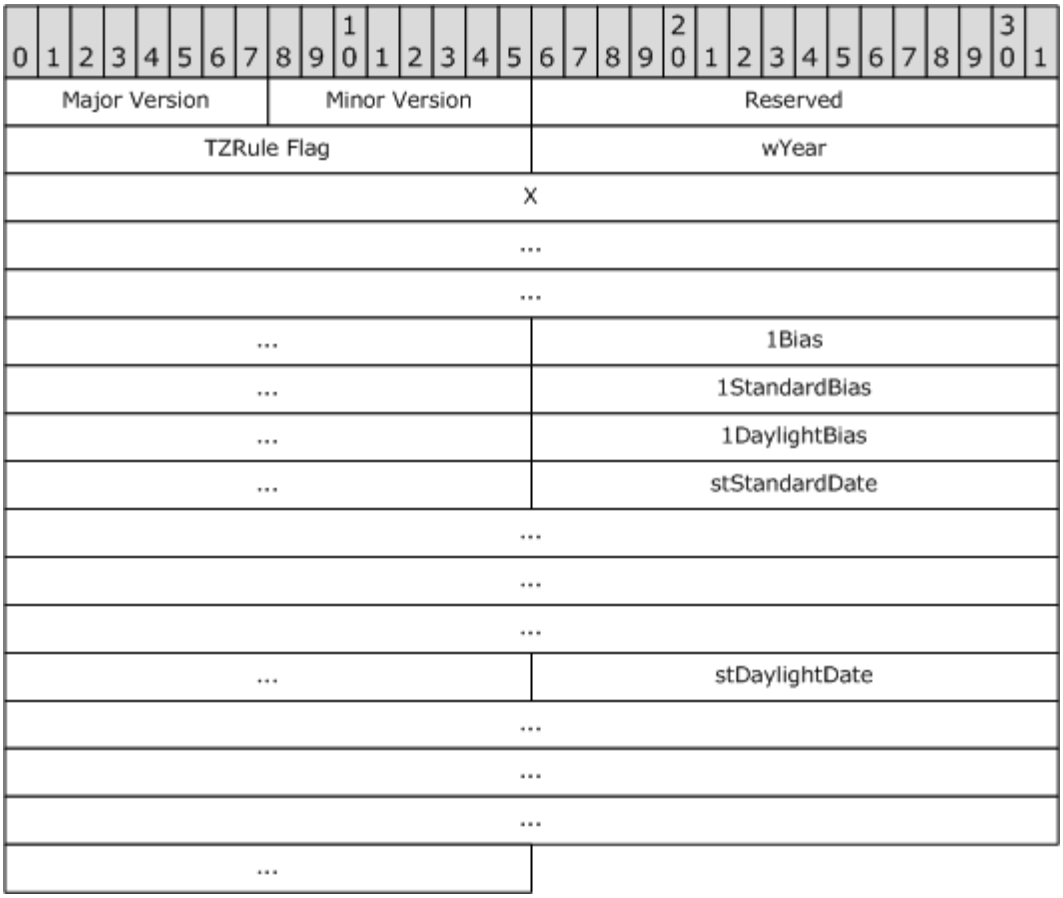
cRules (2 bytes): This WORD property represents the count of *TZRules*. Minimum count is 1; the maximum count is 1024.

TZRules [1..cRules] (2 bytes): Each TZRule contains information that describes a time zone, including the time zone's offset from UTC and when and how it observes daylight saving time. If more than one TZRule is specified, **rules** are sorted in ascending order by the wYear field. *TZRules* are not aligned to 32-bit boundaries. Each TZRule starts at the next byte after the previous TZRule ends. Section 2.2.1.41.1 shows the structure of TZRule, represented in little-endian byte order.

2.2.1.41.1 TZRule

Type: **PtypBinary**

Each TZRule is represented in the following way:



Major version: This field is set to 0x02.

Minor version: This field is set to 0x01.

Reserved: This field MUST be set to0x003E.

TZRule flags: This field contains individual bit flags that specify information about this TZRule, represented here in little-endian byte order.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6
0	0	0	0	0	0	E	R	0	0	0	0	0	0	0	0	0

R (TZRULE_FLAG_RECUR_CURRENT_TZREG, 0x0001): This flag indicates that this rule is associated with a recurring series.

E (TZRULE_FLAG_EFFECTIVE_TZREG, 0x0002): This flag indicates that this rule is the effective rule.

If this rule represents the time zone rule that will be used to convert to and from UTC, both of these flags are set (for example, the value is 0x0003). If this is not the active time zone rule, neither of these flags are set. These flags are set on exactly one TZRule that is contained in this property, and the flags for all other rules MUST be set to 0.

wYear: **WORD** property that represents the year in which this rule is scheduled to take effect. A rule will remain in effect from January 1 of its *wYear* until January 1 of the next rule's *wYear*. If no rules exist for subsequent years, this rule will remain in effect indefinitely.

X: Unused, MUST be all zeros.

lBias: **LONG** property that represents the time zone's offset in minutes from UTC.

lStandardBias: **LONG** property that represents the offset in minutes from *lBias* during standard time.

lDaylightBias: **LONG** property that represents the offset in minutes from *lBias* during daylight saving time.

stStandardDate: **SYSTEMTIME** structure as specified in [MS-DTYP]. This field contains the date and local time that indicate when to begin using the *lStandardBias*.

If the time zone does not support daylight saving time, the *wMonth* member in the **SYSTEMTIME** structure MUST be zero. If the *wYear* member is not zero, the date is interpreted as an absolute date that only occurs once. If the **wYear** member is zero, the date is interpreted as a relative date that occurs yearly. The *wHour* and *wMinute* members are set to the transition time, the *wDayOfWeek* member is set to the appropriate weekday, and the *wDay* member is set to indicate the occurrence of the day of the week within the month (1 to 5, where 5 indicates the final occurrence during the month if that day of the week does not occur 5 times).

stDaylightDate: **SYSTEMTIME** structure as specified in [MS-DTYP]. This field contains the date and local time that indicate when to begin using the *lDaylightBias*. This property has the same format and constraints as the *stStandardDate* field.

2.2.1.42 PidLidAppointmentTimeZoneDefinitionStartDisplay

Type: **PtypBinary**

Specifies time zone information that indicates the time zone of the PidLidAppointmentStartWhole property. The value of this property is used to convert the start date and time from UTC to this time zone for display purposes. The fields in this BLOB are encoded exactly as specified in section 2.2.1.41, with one exception. For each **TZRule** specified by this property, the *R* flag in the *TZRule flags* field is not set (for example, if the **TZRule** is the effective rule, the value of the field *TZRule flags* MUST be 0x0002; otherwise, it MUST be 0x0000).

2.2.1.43 PidLidAppointmentTimeZoneDefinitionEndDisplay

Type: **PtypBinary**

Specifies time zone information that indicates the time zone of the PidLidAppointmentEndWhole property. The format, constraints, and computation of this property are the same as specified in the PidLidAppointmentTimeZoneDefinitionStartDisplay property.

2.2.1.44 PidLidAppointmentRecur

Type: **PtypBinary**

Specifies the dates and times when a recurring series occurs by using one of the recurrence patterns and ranges specified in this section. The value of this property also contains information about both modified and deleted exceptions and information such as dates, subject, location, and other properties of exceptions. The binary data in this property for **Recurring Calendar objects** is stored

as the **AppointmentRecurrencePattern** structure specified in section 2.2.1.44.5. This property MUST NOT exist on single instance Calendar objects.

The following are some limitations of recurrences:

- Multiple instances can not start on the same day.
- Occurrences can not overlap – specifically, an exception that modifies the start date of an instance in the recurring series can occur only on a date that is sometime after the end of the prior instance and the start of the next instance in the recurring series. The same is true if the prior or next instance in the recurring series are exceptions.<13>

The schedule of a recurring series is determined by its recurrence pattern and range. This section describes the types of **recurrence ranges** and recurrence patterns that are supported by this protocol.

Recurrence Range

The recurrence range identifies how long the event will continue. This protocol supports the following three ranges:

- Ends after a specific number of occurrences
- Ends by a given date
- Continues indefinitely

Recurrence Pattern

The recurrence pattern determines the frequency of the event. The **RecurrencePattern** structure is also used to define **recurring tasks**, as specified in [MS-OXOTASK].

The following table lists the types of recurrences that are supported by this protocol.

Recurrence type	Description	Example
Daily recurrence	Schedules events according to one of the following patterns: ▪ Every n number of days. ▪ Every weekday.	An event that repeats every three days, starting on Monday April 30, 2007, and continuing through Friday June 8, 2007.
Weekly recurrence	Schedules events according to the following pattern: ▪ Every n weeks on one or more particular days of the week.	An event repeats every two weeks, on Tuesdays, starting on Monday April 30, 2007, and ending after five occurrences.
Monthly recurrence	Schedules events according to one of the following patterns: ▪ On the n day of every month. ▪ On a specific day of the week on the first, second, third, fourth, or last week of every month. For example, the first	An event that repeats on the fourth of every month, effective Monday April 30, 2007, without an end date.

Recurrence type	Description	Example
	Tuesday of the month.	
Every n months recurrence	A combination of the monthly and weekly patterns. An every n months pattern can schedule events according to one of the following patterns: ▪ On the mth day every n months. ▪ On any day of the week on the first, second, third, fourth, or last week every n months. For example, the third Thursday of the month.	An event that occurs on the last Thursday of every two months, effective March 12, 2007, with an end date of December 31, 2007.
Month end recurrence	Schedules events to repeat on the last day of every n months.	An event that repeats on the last day of every month, effective Monday April 30, 2007, without an end date.
Yearly recurrence	Schedules events according to one of the following patterns: ▪ On the mth day of the nth month, of every year. ▪ On any day of the week on the first, second, third, fourth, or last week of the nth month, of every year. ▪ The yearly recurrence pattern is based on a 12-month interval, and therefore uses the monthly recurrence parameters to represent all the yearly recurrences.	A birthday that occurs every June 22, and is an all-day event.

2.2.1.44.1 RecurrencePattern Structure

This structure specifies a recurrence pattern. The fields of this structure are stored in little-endian byte order.

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
ReaderVersion																WriterVersion															
RecurFrequency																PatternType															
CalendarType																FirstDateTime															
...																Period															
...																SlidingFlag															
...																PatternTypeSpecific (variable)															
EndType																															
OccurrenceCount																															
FirstDOW																															
DeletedInstanceCount																															
DeletedInstanceDates[1..DeletedInstanceCount]																															
ModifiedInstanceCount																															
ModifiedInstanceDates[1..ModifiedInstanceCount]																															
StartDate																															
EndDate																															

ReaderVersion: This field MUST be set to 0x3004.

WriterVersion: This field MUST be set to 0x3004.

RecurFrequency: The *RecurFrequency* field defines the frequency of the recurring series. Valid values are listed in the following table.

RecurFrequency	Value
Daily	0x200A
Weekly	0x200B
Monthly	0x200C
Yearly	0x200D

PatternType: This field defines the type of recurrence pattern. The following table lists the valid recurrence pattern types.<14>

Name	Value	Description
Day	0x0000	The event has a daily recurrence.
Week	0x0001	The event has a weekly recurrence.
Month	0x0002	The event has a monthly recurrence.

Name	Value	Description
MonthNth	0x0003	The event has an every nth month pattern.
HjMonthNth	0x000B	The event has an every nth month pattern in the Hijri calendar. For this PatternType, the CalendarType MUST be set to 0x0000.
HjMonthEnd	0x000C	The event has a month end recurrence in the Hijri calendar. For this PatternType, the CalendarType MUST be set to 0x0000.

CalendarType: This field defines the type of calendar that is used. The following table lists the acceptable values for the calendar type.<15>

Name	Value	Description
Default	0x0000	The default value for the calendar type is Gregorian. If the PatternType is HjMonth, HjMonthNth, or HjMonthEnd, and the CalendarType is Default, this recurrence uses the Hijri calendar.
CAL_GREGORIAN	0x0001	Gregorian (localized) calendar
CAL_GREGORIAN_US	0x0002	Gregorian (U.S.) calendar
CAL_JAPAN	0x0003	Japanese Emperor Era calendar
CAL_TAIWAN	0x0004	Taiwan calendar
CAL_KOREA	0x0005	Korean Tangun Era calendar
CAL_HIJRI	0x0006	Hijri (Arabic Lunar) calendar
CAL_THAI	0x0007	Thai calendar
CAL_HEBREW	0x0008	Hebrew lunar calendar
CAL_GREGORIAN_ME_FRENCH	0x0009	Gregorian Middle East French calendar
CAL_GREGORIAN_ARABIC	0x000A	Gregorian Arabic calendar
CAL_GREGORIAN_XLIT_ENGLISH	0x000B	Gregorian transliterated English calendar
CAL_GREGORIAN_XLIT_FRENCH	0x000C	Gregorian transliterated French calendar
CAL_LUNAR_JAPANESE	0x000E	Japanese lunar calendar
CAL_CHINESE_LUNAR	0x000F	Chinese lunar calendar
CAL_SAKA	0x0010	Saka Era calendar
CAL_LUNAR_KOREAN	0x0014	Korean lunar calendar

FirstDateTime: This field has a different value, depending on the RecurFrequency field. The following table shows how the value of this field is computed, for each recurrence type.

Recurrence type	How calculated
Daily	The value of the *FirstDateTime* field is a numerical value of *StartDate* modulo

Recurrence type	How calculated
Recurrence<16>	*Period*.
Weekly Recurrence<17>	This value is calculated as follows: Find the first *FirstDOW* before *StartDate*. Calculate the number of minutes between midnight that day and midnight, January 1, 1601. Compute the value of *Period* multiplied by 10080, which is the number of minutes in a week. Take the value computed in step 2 modulo the value computed in step 3.
Monthly or Yearly Recurrence<18>	This value is calculated as follows: Find the first day of the month of *StartDate*. Determine **MinimumDate**. For Gregorian calendars, this is midnight, January 1, 1601.For non-Gregorian calendars, this is the first day of the calendar's year that falls in the Gregorian year of 1601. For example, if the **CalendarType** is CAL_HEBREW, the first day of that calendar's year that falls in the Gregorian year of 1601 is 1/1/5362, which is the Gregorian date of 9/27/1601. Calculate the number of calendar months between midnight of the days calculated in step 1 and step 2. Take that value modulo *Period*. Add that number of months to the **MinimumDate**, as determined in step 2. Calculate the number of minutes between midnight that day and midnight, January 1, 1601.

Period: This field is the interval at which the meeting pattern specified in PatternTypeSpecific field repeats. The Period value MUST be between 0 (zero) and the MaximumRecurrenceInterval, which is 999 days for daily recurrences, 99 weeks for weekly recurrences, and 99 months for monthly recurrences. The following table lists the values for this field based on recurrence type.

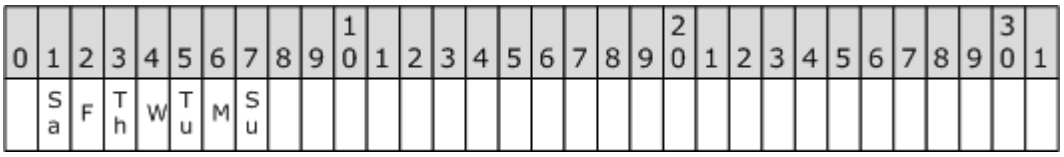
Recurrence type	Value
Daily recurrence	The period is stored as the minutes in whole number of days. For example, to define a recurrence that occurs every two days, the *Period* field is set to 0x00000B40, which equates to 2880 minutes, or two days.
Weekly recurrence	The period is stored in weeks. For example, if the *Period* field is set to 0x00000002, the meeting occurs every two weeks.
Monthly or yearly recurrence	The period is stored in months. If the recurrence is a yearly recurrence, *Period* MUST be set to 12.

SlidingFlag: This field is only used for scheduling tasks; otherwise the value MUST be 0 (zero). For more details about sliding tasks, see [MS-OXOTASK].

PatternTypeSpecific: Specifies the details of the recurrence type and has a different structure, depending on the PatternType. The structure of this field varies based on the recurrence pattern as follows:

For a Daily recurrence pattern (PatternType 0x0000), PatternTypeSpecific has no value and is 0 (zero) bytes. In other words, PatternTypeSpecific is not included in the BLOB when PatternType is 0x0000.

For a Weekly recurrence pattern (PatternType 0x0001), the structure of PatternTypeSpecific is as follows:



Su (0x00000001): The event occurs on Sunday.

M (0x00000002): The event occurs on Monday.

Tu (0x00000004): The event occurs on Tuesday.

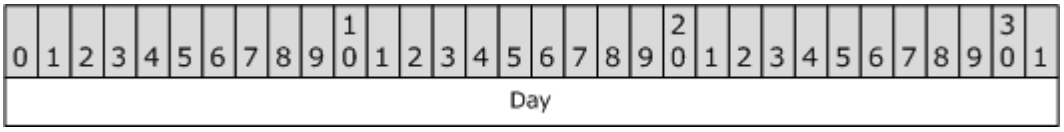
W (0x00000008): The event occurs on Wednesday.

Th (0x00000010): The event occurs on Thursday.

F (0x00000020): The event occurs on Friday.

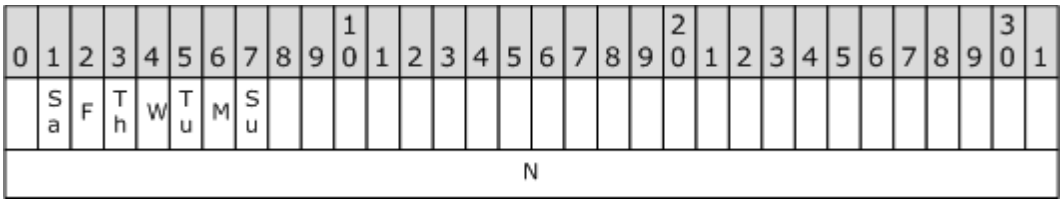
Sa (0x00000040): The event occurs on Saturday.

For the Month, MonthEnd, HjMonth, or HjMonthEnd recurrence pattern (*PatternType* 0x0002, 0x0004, 0x000A, or 0x000C, respectively), the structure of *PatternTypeSpecific* is as follows:



Day: The day of the month on which the recurrence falls.

For the MonthNth or HjMonthNth recurrence pattern (*PatternType* 0x0003 or 0x000B, respectively), the structure of *PatternTypeSpecific* is as follows:



Su (0x00000001): The event occurs on Sunday.

M (0x00000002): The event occurs on Monday.

Tu (0x00000004): The event occurs on Tuesday.

W (0x00000008): The event occurs on Wednesday.

Th (0x00000010): The event occurs on Thursday.

F (0x00000020): The event occurs on Friday.

Sa (0x00000040): The event occurs on Saturday.

If the event occurs on a weekday, the bits M, Tu, W, Th, F, and Sa are set.

If the event occurs on a weekend, the bits Sa and Su are set.

N: The occurrence of the recurrence's days in each month in which the recurrence falls. It can take one of the values listed in the following table.

Name	Value	Description
First	0x00000001	The recurrence falls on the first occurrence of the days specified in every month.
Second	0x00000002	The recurrence falls on the second occurrence of the days specified in every month.
Third	0x00000003	The recurrence falls on the third occurrence of the days specified in every month.
Fourth	0x00000004	The recurrence falls on the fourth occurrence of the days specified in every month.
Last	0x00000005	The recurrence falls on the last occurrence of the days specified in every month.

For example:

- If an event occurs on the last weekday of every two months, the two fields of the *PatternTypeSpecific* field are set to 0x0000003E and 0x00000005.
- If an event occurs on the first weekday of every two months, the two fields of the *PatternTypeSpecific* field are set to 0x0000003E and 0x00000001.
- If an event occurs on the last weekend day of every one month, the two fields of the *PatternTypeSpecific* field are set to 0x00000041and 0x00000005.
- If an event occurs on the first weekend day of every one month, the two fields of the *PatternTypeSpecific* field are set to 0x00000041 and 0x00000001.

EndType: The ending type for the recurrence. This field MUST be set to one of the values listed in the following table.

Recurrence range type	Value
End after date	0x00002021
End after N occurrences	0x00002022
Never end	SHOULD be 0x00002023 but can be 0xFFFFFFFF

OccurrenceCount: The number of occurrences in a recurrence.

When the *EndType* of the pattern is "End after date", this value always has to be computed. Although the value of this field is always set, its value has no meaning on a recurring series that has no end date.<19>

FirstDOW: The first day of the calendar week. The default value is Sunday (0x00000000). This field MUST be set to one of the values listed in the following table.

Day	Value
Sunday	0x00000000

Day	Value
Monday	0x00000001
Tuesday	0x00000002
Wednesday	0x00000003
Thursday	0x00000004
Friday	0x00000005
Saturday	0x00000006

DeletedInstanceCount: This field specifies the number of deleted instances in this recurrence. It is the count of the array of *DeletedInstanceDates*.

DeletedInstanceDates: This field is the array of the original instance date of deleted instances. There is exactly one element for each deleted instance and every deleted instance is represented in this array. Every modified instance also has to have an entry in this array. Deleted instances for which there is no corresponding *ModifiedInstanceDate* imply that they have been completely removed from the pattern.

The count of these instances MUST be equal to the *DeletedInstanceCount* field. Each *DeletedInstanceDate* is stored as the number of minutes between midnight of the specified day and midnight, January 1, 1601, in the time zone specified by PidLidTimeZoneStruct. The values in this list are ordered from earliest to latest. There SHOULD NOT<20> be duplicate entries in this list.

ModifiedInstanceCount: This field specifies the number of positive exceptions for this recurrence. It is the count of the array of *ModifiedInstanceDates*. The value of this field MUST be less than or equal to *DeletedInstanceCount*.

ModifiedInstanceDates: This field is the array of the dates of the modified instances. There is exactly one element for each modified instance and every modified instance has to be represented in this array. Every modified instance has to also have an entry in the array of *DeletedInstanceDates* of the original instance dates.

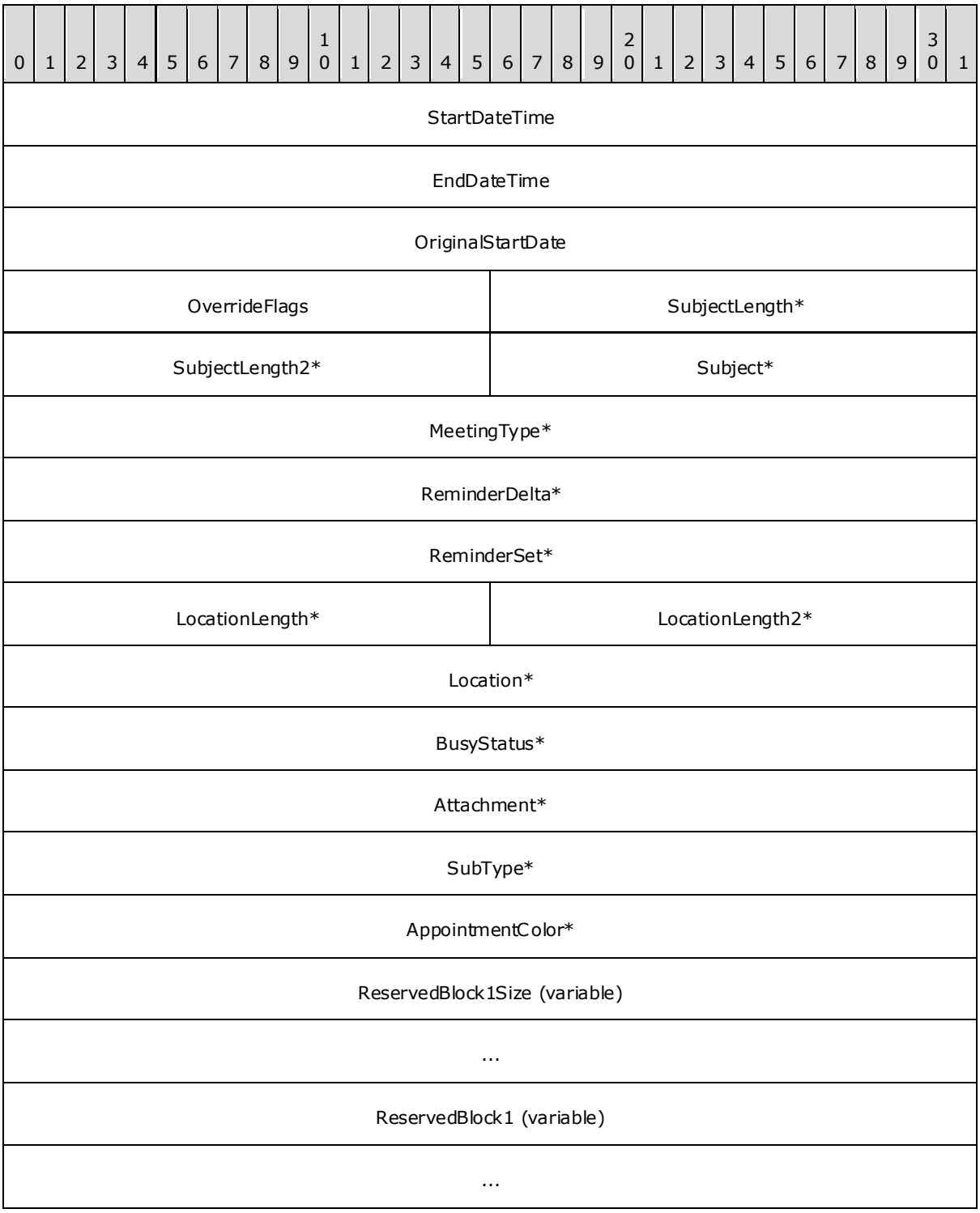
The count of the array MUST be equal to the *ModifiedInstanceCount* field. Each *ModifiedInstanceDate* is stored as the number of minutes between midnight of the specified day and midnight, January 1, 1601, in the time zone specified by PidLidTimeZoneStruct. The values in this list are ordered from earliest to latest. There SHOULD NOT<21> be duplicate entries in this list.

StartDate: The date of the first occurrence. It is stored as the number of minutes between midnight of the specified day and midnight, January 1, 1601.

EndDate: The ending date for the recurrence. It is stored as the number of minutes between midnight of the specified day and midnight, January 1, 1601. When the recurrence range type is "End after N occurrences", this value is calculated as the end date.

If the recurrence does not have an end date, *EndDate* MUST be set to 0x5AE980DF.

2.2.1.44.2 ExceptionInfo Structure



StartDateTime (4 bytes): The start time of the exception in local time in minutes since midnight, January 1, 1601.

EndDateTime (4 bytes): The end time of the exception in local time in minutes since midnight, January 1, 1601.

OriginalStartDate (4 bytes): The original starting time of the exception in local time in minutes since midnight, January 1, 1601.

OverrideFlags (2 bytes): A bit field that specifies what data is present in the *PropertyData* field, which indicates that the exception has a different value than the recurring series. The following table summarizes the valid flags for this field.

Flag	Value	Comments
ARO_SUBJECT	0x0001	Indicates that the *Subject*, *SubjectLength*, and *SubjectLength2* fields are present.
ARO_MEETINGTYPE	0x0002	Indicates that the *MeetingType* field is present.
ARO_REMINDERDELTA	0x0004	Indicates that the *ReminderDelta* field is present.
ARO_REMINDER	0x0008	Indicates that the *ReminderSet* field is present.
ARO_LOCATION	0x0010	Indicates that the Location, LocationLength, and LocationLength2 fields are present.
ARO_BUSYSTATUS	0x0020	Indicates that the *BusyStatus* field is present.
ARO_ATTACHMENT	0x0040	Indicates that the *attachment* field is valid.
ARO_SUBTYPE	0x0080	Indicates that the *SubType* field is present.
ARO_APPTCOLOR <22>	0x0100	This flag is reserved and MUST NOT be set.
ARO_EXCEPTIONAL_BODY	0x0200	Indicates that the Exception Embedded Message object has the PidTagRtfCompressed **property set** on it. See [MS-OXCMSG] section 2.2.1.20.3 for more details about PidTagRtfCompressed.

SubjectLength* (2 bytes): The number of bytes of the *Subject* field plus 1.

* = The presence of this field is conditional based on the value of the *OverrideFlags* field. For more information, see *OverrideFlags* later in this section.

This field is only present if the ARO_SUBJECT flag is set in the *OverrideFlags* field.

SubjectLength2* (2 bytes): The number of bytes of the *Subject* field.

Subject* (2 bytes): A non-null-terminated, non- Unicode string that is the value of the PidTagNormalizedSubject property in the Exception Embedded Message object.

This field is only present if the ARO_SUBJECT flag is set in the *OverrideFlags* field.

MeetingType* (4 bytes): The value of the PidLidAppointmentStateFlags property in the Exception Embedded Message object. For possible values, see section 2.2.1.10. This field is only present if the ARO_MEETINGTYPE flag is set in the *OverrideFlags* field.

ReminderDelta* (4 bytes): The value for the PidLidReminderDelta property (as specified in [MS-OXORMDR]) in the Exception Embedded Message object. This field is only present if the ARO_REMINDERDELTA flag is set in the *OverrideFlags* field.

ReminderSet* (4 bytes): The value for the PidLidReminderSet property (as specified in [MS-OXORMDR]) in the Exception Embedded Message object. This field is only present if the ARO_REMINDER flag is set in the *OverrideFlags* field.

LocationLength* (2 bytes): The number of bytes of the *Location* field plus 1.

This field is only present if the ARO_LOCATION flag is set in the *OverrideFlags* field.

LocationLength2* (2 bytes): The number of bytes of the Location field. This field is only present if the ARO_LOCATION flag is set in the *OverrideFlags* field.

Location* (4 bytes): A non-Unicode string that is the value of the PidLidLocation property in the Exception Embedded Message object. This field is only present if the ARO_LOCATION flag is set in the *OverrideFlags* field.

BusyStatus* (4 bytes): The value for the PidLidBusyStatus property in the Exception Embedded Message object. For possible values, see section 2.2.1.2.This field is only present if the ARO_BUSYSTATUS flag is set in the *OverrideFlags* field.

Attachment* (4 bytes): This value specifies whether or not the Exception Embedded Message object contains **attachments**. The value will be 0x00000001 if attachments are present, and 0x00000000 otherwise. This field is only present if the ARO_ATTACHMENTS flag is set in the *OverrideFlags* field.

SubType* (4 bytes): The value for the PidLidAppointmentSubType property in the Exception Embedded Message object. For possible values, see section 2.2.1.9. This field is only present if the ARO_SUBTYPE flag is set in the *OverrideFlags* field.

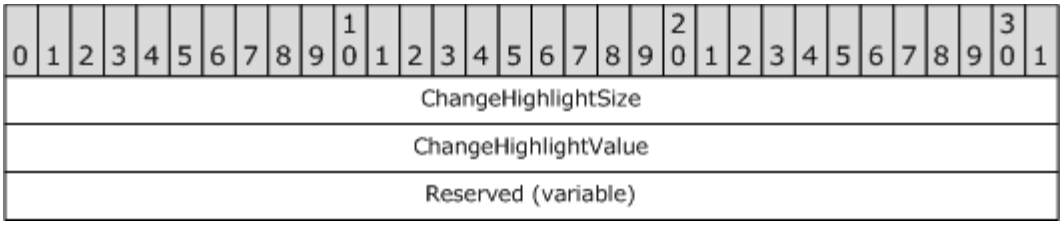
AppointmentColor* (4 bytes): Reserved. This field MUST not be read from or written to.

ReservedBlock1Size (variable): The size of the *ReservedBlock1* field. This field MUST be set to 0 (zero).

ReservedBlock1 (variable): Reserved.

2.2.1.44.3 ChangeHighlight Structure

This field is only present if *WriterVersion2* is greater than or equal to 0x00003009.



ChangeHighlightSize: The size of the *ChangeHighlightValue* and *Reserved* fields combined.

ChangeHighlightValue: The value for the PidLidChangeHighlight property in the Exception Embedded Message object.

Reserved: Reserved.<23>

ReservedBlockEE1Size: The size of the *ReservedBlockEE1* field that follows. This MUST be 0 (zero).

ReservedBlockEE1: Reserved.

StartDateTime: The start time of the exception in local time in minutes since midnight, January 1, 1601.

This field is not present unless either the ARO_SUBJECT or ARO_LOCATION flags are set in the OverrideFlags field of the ExceptionInfo structure.

EndDateTime: The end time of the exception in local time in minutes since midnight, January 1, 1601.

This field is not present unless either the ARO_SUBJECT or ARO_LOCATION flags are set in the *OverrideFlags* field of the *ExceptionInfo* structure.

OriginalStartDate: The original start date of the exception in local time in minutes since midnight, January 1, 1601. This field is not present unless either the ARO_SUBJECT or ARO_LOCATION flags are set in the *OverrideFlags* field of the **ExceptionInfo** structure.

WideCharSubjectLength: The count of Unicode characters in the *WideCharSubject* field. This field is only present if the ARO_SUBJECT flag is set in the *OverrideFlags* field of the **ExceptionInfo** structure.

WideCharSubject: The Unicode string value for the exception's PidTagNormalizedSubject property. Note that *WideCharSubject* is not null-terminated. This field is only present if the ARO_SUBJECT flag is set in the *OverrideFlags* field of the **ExceptionInfo** structure.

WideCharLocationLength: The count of Unicode characters in the *WideCharLocation* field.

This field is only present if the ARO_LOCATION flag is set in the *OverrideFlags* field of the **ExceptionInfo** structure.

WideCharLocation: The Unicode string value for the PidLidLocation property in the Exception Embedded Message object. Note that *WideCharLocation* is not null-terminated. This field is only present if the ARO_LOCATION flag is set in the *OverrideFlags* field of the **ExceptionInfo** structure.

ReservedBlockEE2Size: The size of the *ReservedBlockEE2* field that follows. This field is not present unless either the ARO_SUBJECT or ARO_LOCATION flags are set in the *OverrideFlags* field of the **ExceptionInfo** structure. This field MUST be 0.

ReservedBlockEE2: Reserved. This field is not present unless either the ARO_SUBJECT or ARO_LOCATION flags are set in the *OverrideFlags* field of the **ExceptionInfo** structure.

ReservedBlock2Size: The size of the *ReservedBlock2* field that follows. This field MUST be 0.

ReservedBlock2: Reserved.

2.2.1.44.4 ExtendedException Structure

There is one **ExtendedException** structure per **ExceptionInfo** structure, and each one MUST be in the same order as its corresponding **ExceptionInfo** structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ChangeHighlight (variable)^																															
ReservedBlockEE1Size																															
ReservedBlockEE1 (variable)																															
StartDateTime*																															
EndDateTime*																															
OriginalStartDate*																															
WideCharSubjectLength*																				WideCharSubject (variable)*											
WideCharLocationLength*																				WideCharLocation (variable)*											
ReservedBlockEE2Size*																															
ReservedBlockEE2 (variable)																															

^ = This field is only present if the *WriterVersion2* field is greater than or equal to 0x00003009.

* = The presence of this field is conditional based on the value of the *OverrideFlags* field. For more information, see *OverrideFlags* earlier in this section.

2.2.1.44.5 AppointmentRecurrencePattern Structure

This structure specifies a recurrence pattern for a Calendar object, including information about exception property values. The fields of this structure are stored in little-endian byte order.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
RecurrencePattern (variable)																															
ReaderVersion2																															
WriterVersion2																															
StartTimeOffset																															
EndTimeOffset																															
ExceptionCount																				ExceptionInfo (variable) [1..ExceptionCount]											
ReservedBlock1Size																															
ReservedBlock1 (variable)																															
ExtendedException (variable) [1..ExceptionCount]																															
ReservedBlock2Size																															
ReservedBlock2 (variable)																															

RecurrencePattern: This field is a **RecurrencePattern** structure that defines the recurrences. For details, see section 2.2.1.44.1

ReaderVersion2: This value MUST be set to 0x00003006.

WriterVersion2: This value SHOULD be set to 0x00003009, but can be set to 0x00003008. The value of this field affects the format of the *ExtendedException* field.

StartTimeOffset: The number of minutes since midnight after which each occurrence starts. For example, the value for midnight is 0 (zero) and the value for 12:00 P.M. is 720.

EndTimeOffset: The number of minutes since midnight after which each occurrence ends. For example, the value for midnight is 0 (zero) and the value for 12:00 P.M. is 720.

ExceptionCount: This field is the count of **ExceptionInfo** structures. This is also the count of **ExtendedException** structures. This MUST be the same value as the *ModifiedInstanceCount*.

2.2.1.45 PidLidRecurrenceType

Type: **PtypInteger32**

Specifies the recurrence type of the recurring series by using one of the values listed in the following table.

Status	Value	Description
rectypeNone	0	A single-instance appointment.
rectypeDaily	1	A daily recurrence pattern.
rectypeWeekly	2	A weekly recurrence pattern.
rectypeMonthly	3	A monthly recurrence pattern.
rectypeYearly	4	A yearly recurrence pattern.

2.2.1.46 PidLidRecurrencePattern

Type: **PtypString**

Specifies a description of the recurrence pattern of the Calendar object. This property is not required, but if set, it is set to a description of the recurrence specified by the PidLidAppointmentRecur property.

2.2.1.47 PidLidLinkedTaskItems

Type: **PtypMultipleBinary**

Specifies a list of the PidTagEntryId properties of **task objects** [MS-OXOTASK] that are related to the Calendar object. This property is not required. <24>

2.2.1.48 PidLidMeetingWorkspaceUrl

Type: **PtypString**

Specifies the **URL** of the **Meeting Workspace**, as specified in [MS-MEETS], that is associated with a Calendar object. This property is not required.

2.2.1.49 PidTagIconIndex

Type: **PtypInteger32**

The value of this property indicates that an icon is used with the object. It SHOULD<25> be set to one of the following, but can be -1.

Description	Value	Used by
Single-instance appointment	0x00000400	Appointment object
Recurring appointment	0x00000401	Appointment object
Single-instance meeting	0x00000402	Meeting object
Recurring meeting	0x00000403	Meeting object
Meeting request/full update	0x00000404	Meeting Request object, Meeting Update object
Accept	0x00000405	Meeting Response object
Decline	0x00000406	Meeting Response object
Tentatively accept	0x00000407	Meeting Response object
Cancellation	0x00000408	Meeting Cancellation object
Informational update	0x00000409	Meeting Update object
Forward notification	0x0000040b	Meeting Forward Notification object

2.2.1.50 Deprecated properties

The following properties are deprecated and SHOULD NOT be written by clients or servers <26>. If PidLidConferencingCheck is set to FALSE, all the properties in this section are ignored. These properties are only to be set on Calendar objects and meeting-related objects.

2.2.1.50.1 PidLidConferencingCheck

Type: **PtypBoolean**

This property indicates that this meeting is one of the following types:

- "Windows Media Services"
- "Windows NetMeeting"
- "Exchange Conferencing"

If this property is set, PidLidConferencingType is also to be set. This property is set to TRUE only on Meeting objects or meeting-related objects.

2.2.1.50.2 PidLidConferencingType

Type: **PtypInteger32**

This property specifies the type of the meeting. The value of this property MUST be set to one of the values listed in the following table.

Type of meeting	Value

Type of meeting	Value
Windows Netmeeting	0x00000000
Windows Media Services	0x00000001
Exchange Conferencing	0x00000002

2.2.1.50.3 PidLidDirectory

Type: **PtypString**

This property specifies the directory server to be used with NetMeeting.

2.2.1.50.4 PidLidAllowExternalCheck

Type: **PtypBoolean**

This property MUST be set to TRUE.

2.2.1.50.5 PidLidOrganizerAlias

Type: **PtypString**

This property specifies the e-mail address of the organizer.

2.2.1.50.6 PidLidCollaborateDoc

Type: **PtypString**

This property specifies the **document** to be launched when the user joins the meeting. This property is valid only when PidLidConferencingType has the value 0x00000000.

2.2.1.50.7 PidLidNetShowUrl

Type: **PtypString**

This property specifies the URL to be launched when the user joins the meeting. This property is valid only when the PidLidConferencingType property has the value 0x00000001 or 0x00000002.

For meetings with 0x00000001 as the value of PidLidConferencingType, this is a user-supplied URL. For meetings with 0x00000002 as the value of PidLidConferencingType, this URL is generated as follows:

- For each **Bcc recipient** of a Meeting Request object, open the associated folder of the Calendar folder in the recipient's mailbox.
- Find the message the PidTagMessageClass property for which has a value of "EXCH_CONFERENCE." If the message is not found, move on to the next Bcc recipient. If the message is found, open it and get its PidTagLocation property.
- Append the base64-encoded value of the PidLidGlobalObjectId property of the Meeting object.
- Append the string "&p=" followed by the value of the PidLidOnlinePassword property.
- Finally, convert the string to Unicode.

If there are multiple Exchange Conferencing mailboxes in the BCC field, the value that is calculated by using the last mailbox is used.

2.2.1.50.8 PidLidOnlinePassword

Type: **PtypString**

This property specifies the password for a meeting on which the property PidLidConferencingType has the value 0x00000002. If set, this string is a maximum of 255 characters, not including NULL.

2.2.2 Calendar Object

This section specifies properties that are specific to Calendar objects. <27>Unless otherwise specified, these properties are to always exist.

2.2.2.1 PidTagMessageClass

Type: **PtypString8**

The value of this property MUST be "IPM. appointment" or be prefixed with "IPM.appointment".

2.2.2.2 PidLidSideEffects

Type: **PtypInteger32**

The possible flag values of this property are specified in [MS-OXCMSG]. All Calendar objects SHOULD<28> include the following flags:

seOpenToDelete

seOpenToCopy

seOpenToMove

seCoerceToInbox

seOpenForCtxMenu

2.2.2.3 PidLidFExceptionalAttendees

Type: **PtypBoolean**

A value of TRUE for this property indicates that it is a Recurring Calendar object with one or more exceptions, and at least one of the Exception Embedded Message objects has at least one **RecipientRow**. A value of FALSE, or the absence of this property, indicates that the Calendar object either has no exceptions, or that none of the Exception Embedded Message objects has **RecipientRows**.<29>

2.2.3 Meeting Object

This section specifies the properties that are specific to Meeting objects. These properties have no meaning for Appointment objects. <30> Unless otherwise specified, these properties are to always exist.

2.2.3.1 PidLidAppointmentSequenceTime

Type: **PtypTime**

The value of this property on the organizer'sMeeting object indicates the date and time at which the property PidLidAppointmentSequence was last modified. The value is specified in UTC.

2.2.3.2 PidLidAppointmentLastSequence

Type: **PtypInteger32**

The value of this property indicates to the organizer the last sequence number that was sent to any attendee. For details about when and how a client increments the sequence number, see section 3.1.5.4. This property has no meaning for an attendee.

2.2.3.3 PidLidAppointmentReplyTime

Type: **PtypTime**

The value of this property on the attendee's Meeting object specifies the date and time at which the attendee responded to a received meeting request or Meeting Update object. The value is specified in UTC.

2.2.3.4 PidLidFInvited

Type: **PtypBoolean**

This property indicates whether invitations have been sent for the meeting that this Meeting object represents. A value of FALSE, or the absence of this property, indicates that a Meeting Request object has never been sent. A value of TRUE indicates that a Meeting Request object has been sent. After this value is set to TRUE on a Meeting object, it MUST NOT be changed.

2.2.3.5 PidLidAppointmentReplyName

Type: **PtypString**

This property on the attendee's Meeting object specifies the user who last replied to the meeting request or **meeting update**. This property is set only for a delegator when a delegate responded. The value is equal to the PidTagMailboxOwnerName property for the delegate's **store**. This property has no meaning for the organizer. For details about PidTagMailboxOwnerName, see [MS-OXCSTOR].

2.2.3.6 PidLidAppointmentProposalNumber

Type: **PtypInteger32**

This property specifies the number of attendees who have sent **counter proposals** that have not been accepted or rejected by the organizer.

2.2.3.7 PidLidAppointmentCounterProposal

Type: **PtypBoolean**

This property indicates to the organizer that there are counter proposals that have not been accepted or rejected (by the organizer). This property has no meaning for an attendee.

2.2.3.8 PidLidAutoFillLocation

Type: **PtypBoolean**

A value of TRUE for this **Boolean** property on the organizer's Meeting object indicates that the value of the PidLidLocation property is set to the PidTagDisplayName property from the **RecipientRow** that represents a resource.<31>For more details about **RecipientRow**, see [MS-OXCMSG].

2.2.3.9 RecipientRow Properties

The Meeting object has one **RecipientRow** (as specified in [MS-OXCMSG]) for each sendable attendee. In addition, a **RecipientRow** can exist for the organizer of the Meeting object. unsendable attendees do not have a corresponding **RecipientRow**, but SHOULD have a row in the PidLidAppointmentUnsendableRecipients property (see section 2.2.1.25). The appointment and Meeting object protocol defines properties that can be set in the "Extra properties" section of **RecipientRows**. These are listed in the following sections.

2.2.3.9.1 PidTagRecipientFlags

Type: **PtypInteger32**

Specifies a bit field that describes the recipient status. This property is not required. The following are the individual flags that can be set:

- *S* (recipSendable, 0x00000001): The recipient is a sendable attendee. This flag is used only in the PidLidAppointmentUnsendableRecipients property.
- *O* (recipOrganizer, 0x0000002): The **RecipientRow** on which this flag is set represents the meeting organizer.
- *ER* (recipExceptionalResponse, 0x00000010): Indicates that the attendee gave a response for the exception on which this RecipientRow resides. This flag is used only in a **RecipientRow** of an Exception Embedded Message object of the organizer's Meeting object.
- *ED* (recipExceptionalDeleted, 0x00000020): Indicates that although the **RecipientRow** exists, it is treated as if the corresponding recipient does not exist. This flag is used only in a **RecipientRow** of an Exception Embedded Message object of the organizer's Meeting object.
- *X*: MUST NOT be set (reserved, 0x00000040) <32>.
- *X*: MUST NOT be set (reserved, 0x00000080) <33>.
- G: (recipOriginal, 0x00000100): Indicates that the recipient is an original attendee. This flag is used only in the PidLidAppointmentUnsendableRecipients property.
- *X*: (reserved, 0x00000200) <34>.

2.2.3.9.2 PidTagRecipientTrackStatus

Type: **PtypInteger32**

The value of this property indicates the response status that is returned by the attendee. If this value is not set, it is assumed to be respNone. If set, it MUST be one of the following, as specified in section 2.2.1.11:

- respNone

- respAccepted
- respDeclined
- respTentative

2.2.3.9.3 PidTagRecipientTrackStatusTime

Type: **PtypTime**

This property indicates the date and time at which the attendee responded. The value is specified in UTC.

2.2.3.9.4 PidTagRecipientProposed

Type: **PtypBoolean**

A value of TRUE for this property indicates that the attendee proposed a new date and/or time. A value of FALSE, or the absence of this property, means either that the attendee did not yet respond, or that the most recent response from the attendee did not include a new date/time proposal. This value can not be TRUE for attendees in a recurring series.

2.2.3.9.5 PidTagRecipientProposedStartTime

Type: **PtypTime**

When the value of the PidTagRecipientProposed property is set to TRUE, the value of this property indicates the value requested by the **attendee** to set as the value of the PidLidAppointmentStartWhole property for the single instance meeting objector **exception object**.

2.2.3.9.6 PidTagRecipientProposedEndTime

Type: **PtypTime**

When the value of the PidTagRecipientProposed property is set to TRUE, the value of this property indicates the value requested by the attendee to set as the value of the PidLidAppointmentEndWhole property for the single instance Meeting object or Exception object.

2.2.3.9.7 Recipient Type

Type: **PtypInteger32**

This property is specified in [MS-OXCMSG]. The appropriate value is set as the recipient type for each **RecipientRow** in the Meeting object. The following table lists the appropriate values for the recipient type.

attendee type	recipient type
organizer	0x01
Sendable, required attendee	0x01
Sendable, optional attendee	0x02
Sendable, resource	0x03 (only on the Meeting object in the organizer's Calendar folder)

2.2.4 Meeting-Related Objects

This section specifies properties that are specific to meeting-related objects. These include meeting request, meeting update, **Meeting Cancellation**, Meeting Response, and Meeting Forward Notification objects. Unless otherwise specified, these properties MUST exist.

2.2.4.1 PidLidSideEffects

Type: **PtypInteger32**

The possible flag values of this property are specified in [MS-OXCMSG]. All Meeting Request objects are to always include the following flags:

seOpenToDelete (0x00000001)

seOpenToCopy (0x00000020)

seOpenToMove (0x00000040)

seCannotUndoDelete (0x00000400)

seCannotUndoCopy (0x00000800)

seCannotUndoMove (0x00001000)

2.2.4.2 PidLidAttendeeCriticalChange

Type: **PtypTime**

The value of this property specifies the date and time at which the meeting-related object was sent. The value is specified in UTC. <35>

2.2.4.3 PidLidWhere

Type: **PtypString**

The value of this property SHOULD be the same as the value of the PidLidLocation property from the associated Meeting object. <36>

2.2.4.4 PidLidServerProcessed

Type: **PtypBoolean**

A value of TRUE for this **Boolean** property indicates that the Meeting Request object or Meeting Update object has been processed.

2.2.4.5 PidLidServerProcessingActions

Type: **PtypInteger32**

This property indicates what processing **actions** have been taken on this Meeting Request object or Meeting Update object. The following flags can be set.

Flag	Value
cpsDelegatorWantsCopy	0x00000002

Flag	Value
cpsCreatedOnPrincipal	0x00000010
cpsUpdatedCalItem	0x00000080
cpsCopiedOldProperties	0x00000100
cpsSendAutoResponse	0x00000400
cpsRevivedException	0x00000800
cpsProcessedMeetingForwardNotification	0x00001000

2.2.4.6 PidLidTimeZone

Type: **PtypInteger32**

The value of this property specifies information about the time zone of a recurring meeting. This property is only read if PidLidAppointmentRecur is not set, but PidLidIsRecurring is TRUE and PidLidIsException is FALSE. The lower **WORD** specifies an index into a table that contains time zone information. From the upper **WORD**, only the highest bit is read. If that bit is set, the time zone referenced will not observe daylight saving time; otherwise, the daylight saving time dates listed in the following table will be used<37>.

Index	Standard offset from UTC+12 (international date line) in minutes	Standard date {wMonth, wDayOfWeek, wDay, wHour}	Daylight date {wMonth, wDayOfWeek, wDay, wHour}
0	0	N/A	N/A
1	12*60	{10, 0, 5, 2}	{3, 0, 5, 1}
2	11*60	{9, 0, 5, 2}	{3, 0, 5, 1}
3	11*60	{10, 0, 5, 3}	{3, 0, 5, 2}
4	11*60	{10, 0, 5, 3}	{3, 0, 5, 2}
5	10*60	{9, 0, 5, 1}	{3, 0, 5, 0}
6	11*60	{9, 0, 5, 1}	{3, 0, 5, 0}
7	10*60	{10, 0, 5, 4}	{3, 0, 5, 3}
8	15*60	{2, 0, 2, 2}	{10, 0, 3, 2}
9	16*60	{11, 0, 1, 2}	{3, 0, 2, 2}
10	17*60	{11, 0, 1, 2}	{3, 0, 2, 2}
11	18*60	{11, 0, 1, 2}	{3, 0, 2, 2}
12	19*60	{11, 0, 1, 2}	{3, 0, 2, 2}
13	20*60	{11, 0, 1, 2}	{3, 0, 2, 2}

Index	Standard offset from UTC+12 (international date line) in minutes	Standard date {wMonth, wDayOfWeek, wDay, wHour}	Daylight date {wMonth, wDayOfWeek, wDay, wHour}
14	21*60	{11, 0, 1, 2}	{3, 0, 2, 2}
15	22*60	N/A	N/A
16	23*60	N/A	N/A
17	0*60	{4, 0, 1, 3}	{9, 0, 5, 2}
18	2*60	{3, 0, 5, 3}	{10, 0, 5, 2}
19	(2*60)+30	{3, 0, 5, 3}	{10, 0, 5, 2}
20	3*60	N/A	N/A
21	4*60	N/A	N/A
22	5*60	N/A	N/A
23	(6*60)+30	N/A	N/A
24	8*60	N/A	N/A
25	(8*60)+30	{9, 2, 4, 2}	{3, 0, 1, 2}
26	9*60	N/A	N/A
27	10*60	{9, 0, 3, 2}	{3, 5, 5, 2}
28	(15*60)+30	{11, 0, 1, 0}	{3, 0, 2, 0}
29	13*60	{10, 0, 5, 1}	{3, 0, 5, 0}
30	14*60	{10, 0, 5, 1}	{3, 0, 5, 0}
31	12*60	N/A	N/A
32	15*60	N/A	N/A
33	16*60	N/A	N/A
34	17*60	N/A	N/A
35	17*60	N/A	N/A
36	18*60	N/A	N/A
37	18*60	{10, 0, 5, 2}	{4, 0, 1, 2}
38	19*60	N/A	N/A
39	24*60	N/A	N/A
40	0*60	N/A	N/A
41	1*60	N/A	N/A

Index	Standard offset from UTC+12 (international date line) in minutes	Standard date {wMonth, wDayOfWeek, wDay, wHour}	Daylight date {wMonth, wDayOfWeek, wDay, wHour}
42	2*60	{3, 0, 5, 2}	{10, 0, 1, 2}
43	2*60	N/A	N/A
44	(2*60)+30	N/A	N/A
45	4*60	{9, 0, 2, 2}	{4, 0, 2, 2}
46	6*60	N/A	N/A
47	7*60	N/A	N/A
48	(7*60)+30	N/A	N/A
49	10*60	{9, 4, 5, 2}	{5, 5, 1, 2}
50	10*60	N/A	N/A
51	9*60	{10, 0, 5, 1}	{3, 0, 5, 0}
52	2*60	{3, 0, 5, 2}	{8, 0, 5, 2}
53	2*60	{4, 0, 1, 3}	{10, 0, 5, 2}
54	(2*60)+30	{4, 0, 1, 3}	{10, 0, 5, 2}
55	2*60	{4, 0, 1, 3}	{10, 0, 1, 2}
56	16*60	{3, 6, 2, 23}	{10, 6, 2, 23}
57	4*60	{3, 0, 5, 3}	{10, 0, 5, 2}
58	19*60	{10, 0, 5, 2}	{4, 0, 1, 2}
59	20*60	{10, 0, 5, 2}	{4, 0, 1, 2}

The Standard date and Daylight date columns specify a date in the following format:

{*wMonth*, *wDayOfWeek*, *wDay*, *wHour*}

These values are interpreted as follows:

wMonth :

Value	Meaning
1	January
2	February
3	March
4	April

Value	Meaning
5	May
6	June
7	July
8	August
9	September
10	October
11	November
12	December

wDayOfWeek :

Value	Meaning
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

wDay : Indicates the occurrence of the day of the week within the month (1 to 5, where 5 indicates the final occurrence during the month if that day of the week does not occur 5 times).

wHour : Indicates the hour at which the transition will occur in local time. The member ranges in value from 0 (zero) (12:00 A.M.) to 23 (11:00 P.M.).

If daylight saving time is observed, during the daylight time period, an additional -60 offset is added to the standard offset.

2.2.5 Meeting Request/Update Object

This section specifies the properties that are specific to Meeting Request objects and Meeting Update objects. <38> Unless otherwise specified, these properties are to always exist.

2.2.5.1 PidTagMessageClass

Type: **PtypString8**

The value of this property MUST be "IPM.Schedule.meeting.Request" or be prefixed with "IPM.Schedule.meeting.Request".

2.2.5.2 PidLidChangeHighlight

Type: **PtypInteger32**

Specifies a bit field that indicates how the Meeting object has changed. <39> This property is not required. The following are the individual flags that can be set.

ST (BIT_CH_START, 0x00000001): The property PidLidAppointmentStartWhole has changed.

ET (BIT_CH_END, 0x00000002): The property PidLidAppointmentEndWhole has changed.

REC (BIT_CH_RECUR, 0x00000004): The recurrence pattern has changed. See the property PidLidAppointmentRecur.

LOC (BIT_CH_LOCATION, 0x00000008): The property PidLidLocation has changed.

SUB (BIT_CH_SUBJECT, 0x00000010): The property PidTagNormalizedSubject has changed.

REQ (BIT_CH_REQATT, 0x00000020): One or more required attendees were added.

OPT (BIT_CH_OPTATT, 0x00000040): One or more optional attendees were added.

B (BIT_CH_BODY, 0x00000080): The body was modified.

RE (BIT_CH_RESPONSE, 0x00000200): Either the property PidTagResponseRequested or the property PidTagReplyRequested has changed.

AP (BIT_CH_ALLOWPROPOSE, 0x00000400): The property PidLidAppointmentNotAllowPropose has changed.

CNF (0x00000800): Deprecated.

REM (0x00001000): Reserved.

OTH (0x08000000): Reserved.

2.2.5.3 PidLidForwardInstance

Type: **PtypBoolean**

A value of TRUE for this property indicates that the Meeting Request object represents an exception to a recurring series, and it was forwarded (even when forwarded by the organizer) rather than being an invitation sent by the organizer. A value of FALSE for this property indicates that the Meeting Request object is not a forwarded instance. This property is not required. <40>

2.2.5.4 PidLidIntendedBusyStatus

Type: **PtypInteger32**

Specifies the value of the PidLidBusyStatus property on the Meeting object in the organizer's calendar at the time the Meeting Request object or Meeting Update object was sent. The allowable values of this property are the same as those for the property PidLidBusyStatus.

2.2.5.5 PidLidMeetingType

Type: **PtypInteger32**

This property indicates the type of Meeting Request object or Meeting Update object. The value of this property MUST be set to one of those listed in the following table.

Property	Value	Description
mtgEmpty	0x00000000	Unspecified.
mtgRequest	0x00000001	Initial meeting request.
mtgFull	0x00010000	Full update.
mtgInfo	0x00020000	Informational update.
mtgOutOfDate	0x00080000	A newer Meeting Request object or Meeting Update object was received after this one. For more details, see section 3.1.5.2.
mtgDelegatorCopy	0x00100000	This is set on the delegator's copy when a delegate will handle meeting-related objects. For more details, see section 3.1.4.6.2.1.

2.2.5.6 PidLidAppointmentMessageClass

Type: **PtypString**

This **String** property indicates the PidTagMessageClass of the Meeting object that is to be generated from the Meeting Request object. The value of this property MUST either be "IPM.appointment" or be prefixed with "IPM.appointment". This property is not required.

2.2.5.7 PidLidOldLocation

Type: **PtypString**

This property indicates the original value of the PidLidLocation property before a meeting update<41>. This property is not required.

2.2.5.8 PidLidOldWhenStartWhole

Type: **PtypTime**

This property indicates the original value of the PidLidAppointmentStartWhole property before a meeting update<42>. This property is not required.

2.2.5.9 PidLidOldWhenEndWhole

Type: **PtypTime**

This property indicates the original value of the PidLidAppointmentEndWhole property before a meeting update<43>. This property is not required.

2.2.5.10 Attachments

A Meeting Request object or Meeting Update object represents a single instance, a recurring series, or an exception. A Meeting Request object or a Meeting Update object for a recurring series can not include any Exception Attachment objects. A separate Meeting Request object or Meeting Update object is to be sent for each exception, even when attendees are invited to both the recurring series and the exceptions.

2.2.5.11 PidLidCalendarType

Type: **PtypInteger32**

When the Meeting Request object represents a recurring series or an exception, this property is the value of the *CalendarType* field from the PidLidAppointmentRecur property. Otherwise, this property is not set and is assumed to be 0 (zero).

2.2.5.12 Best Body Properties

The body of a Meeting Request object is a copy of the body of the Meeting object or Exception Embedded Message object to which it refers, optionally preceded by "downlevel text." The term "downlevel text" refers to extra text that can be added into the body of a Meeting Request object before a copy of the Meeting object body, so that a client that receives the Meeting Request object but does not understand its format will still show the meeting details. Downlevel text is to be separated from the copied Meeting object body with a delimiter, and then the delimiter is to be followed by two blank lines. The following table lists the delimiters. <44>

PidLidCalendarType	Delimiter
CAL_HIJRI	+=+=+=+=+=+=+=+=+=+
CAL_HEBREW	+=+=+=+=+=+=+=+=+=+
CAL_THAI	+=+=+=+=+=+=+=+=+=+
CAL_LUNAR_KOREAN	+=+=+=+=+=+=+=+=+=+
CAL_LUNAR_JAPANESE	+=+=+=+=+=+=+=+=+=+
CAL_CHINESE_LUNAR	+=+=+=+=+=+=+=+=+=+
CAL_SAKA	+=+=+=+=+=+=+=+=+=+
CAL_GREGORIAN	*~*~*~*~*~*~*~*~*~*
Any other value	*~*~*~*~*~*~*~*~*~*

2.2.6 Meeting Response Object

This section specifies the properties that are specific to Meeting Response objects

A Meeting Response object takes the form of one of three types: Accept, Tentatively Accept, or Decline. These properties apply to all response types, except where individually noted. Unless otherwise specified, these properties are to always exist.

2.2.6.1 PidTagMessageClass

Type: **PtypString8**

The value of this property MUST begin with "IPM.Schedule.meeting.Resp" and MUST be appended with either "Pos", ".Tent", or ".Neg", indicating accept, tentatively accept, or decline, respectively.

2.2.6.2 PidTagSubjectPrefix

Type: **PtypString**

The value of this property is a localized string that indicates accept, tentatively accept, or decline, unless the Meeting Response object includes a new date/time proposal, in which case this is to be indicated by the value of this property<45>

2.2.6.3 PidLidAppointmentProposedStartWhole

Type: **PtypTime**

Specifies the proposed value for PidLidAppointmentStartWhole for a counter proposal. This value is specified in UTC.

2.2.6.4 PidLidAppointmentProposedEndWhole

Type: **PtypTime**

Specifies the proposed value for PidLidAppointmentEndWhole for a counter proposal. This value is specified in UTC.

2.2.6.5 PidLidAppointmentProposedDuration

Type: **PtypInteger32**

This property indicates the proposed value for PidLidAppointmentDuration for a counter proposal. If set, it is equal to the number of minutes between PidLidAppointmentProposedStartWhole and PidLidAppointmentProposedEndWhole.

2.2.6.6 PidLidAppointmentCounterProposal

Type: **PtypBoolean**

A value of TRUE for this property indicates that this Meeting Response object is a counter proposal.

2.2.6.7 PidLidIsSilent

Type: **PtypBoolean**

A value of TRUE for this property indicates that the user did not include any text in the body of the Meeting Response object.

2.2.6.8 PidLidPromptSendUpdate

Type: **PtypBoolean**

A value of TRUE for this property indicates that the Meeting Response object was out-of-date when it was received.

2.2.7 Meeting Cancellation Object

This section specifies the properties that are specific to Meeting Cancellation objects. Unless otherwise specified, these properties are to always exist.

2.2.7.1 PidTagMessageClass

Type: **PtypString8**

The value of this property MUST be "IPM.Schedule.meeting.Canceled."

2.2.7.2 PidTagSubjectPrefix

Type: **PtypString**

The value of this property is a localized string that indicates that the meeting was canceled.<46>

2.2.7.3 PidLidIntendedBusyStatus

Type: **PtypInteger32**

The value of this property MUST be set to olFree.

2.2.7.4 PidLidResponseStatus

Type: **PtypInteger32**

The value of this property MUST be set to respNotResponded.

2.2.7.5 PidLidBusyStatus

Type: **PtypInteger32**

The value of this property MUST be set to olFree.

2.2.8 Meeting Forward Notification Object

This section specifies the properties that are specific to Meeting Forward Notification objects. Unless otherwise specified, these properties MUST exist.

2.2.8.1 PidTagMessageClass

Type: **PtypString8**

The value of this property MUST be "IPM.Schedule.meeting.Notification.Forward".

2.2.8.2 PidTagSubjectPrefix

Type: **PtypString**

The value of this property MUST be a localized string that indicates that the object is a Meeting Forward Notification object.

2.2.8.3 PidLidForwardNotificationRecipients

Type: **PtypBinary**

This binary property contains a list of **RecipientRows** that indicate the recipients of a meeting forward. See the PidLidAppointmentUnsendableRecipients property in 2.2.1.25 for the format of this property.

2.2.8.4 PidLidPromptSendUpdate

Type: **PtypBoolean**

A value of TRUE for this property indicates that the Meeting Forward Notification object was out-of-date when it was received.

2.2.9 Exceptions

An exception specifies changes to an instance of a recurring series. Two objects define an exception: the Exception Attachment object and the Exception Embedded Message object. One Exception Attachment object SHOULD<47> exist for each instance listed in the *ModifiedInstanceDates* field of the PidLidAppointmentRecur property on the Calendar object. One Exception Embedded Message object MUST exist for each Exception Attachment object.

The Exception Attachment object is an **Attachment object**, as specified in [MS-OXCMSG], and holds attachment-related information. The Exception Embedded Message object is an **Embedded Message object**, as specified in [MS-OXCMSG], and holds the modifications to the instance. This section specifies the properties that are specific to the Exception Attachment object and the Exception Embedded Message object that make up the exception. Unless otherwise specified, these properties are to always exist.

2.2.9.1 Exception Attachment Object

The Exception Attachment object MUST have the properties listed in the following sections.

2.2.9.1.1 PidTagAttachmentHidden

Type: **PtypBoolean**

This property is specified in [MS-OXCMSG]. The value of this property MUST be TRUE.

2.2.9.1.2 PidTagAttachmentFlags

Type: **PtypInteger32**

This property is specified in [MS-OXCMSG]. The value MUST include the afException (0x00000002) flag.

2.2.9.1.3 PidTagAttachMethod

Type: **PtypInteger32**

This property is specified in [MS-OXCMSG]. The value MUST be afEmbeddedMessage (0x00000005), which indicates that the exception data in PidTagAttachDataObject is an Embedded Message object.

2.2.9.1.4 PidTagExceptionStartTime

Type: **PtypTime**

The value of this property indicates the start date and time of the exception in the local time zone of the computer when the exception is created. This property is informational and can not<48> be relied on for critical information.

2.2.9.1.5 PidTagExceptionEndTime

Type: **PtypTime**

The value of this property indicates the end date and time of the exception in the local time zone of the computer when the exception is created. This property is informational and can not<49>be relied on for critical information.

2.2.9.1.6 PidTagExceptionReplaceTime

Type: **PtypTime**

The value of this property indicates the original date and time at which the instance in the recurrence pattern would have occurred if it were not an exception. This value is specified in UTC<50>.

2.2.9.2 Exception Embedded Message Object

The data stored in the Embedded Message object that is represented by the PidTagAttachDataObject property (see [MS-OXCMSG]) contains properties that are specific to the exception. Any property that is not set on the Exception Embedded Message object is obtained from the recurrence **series**. The following properties SHOULD NOT be set on an Exception Embedded Message object; if they are set, they are not used by the client or server:

- PidLidAppointmentSequence
- PidLidAppointmentSequenceTime
- PidLidAppointmentLastSequence
- PidLidMeetingWorkspaceUrl
- PidLidContacts (see [MS-OXCMSG])
- PidTagSensitivity (see [MS-OXCMSG])
- PidLidPrivate (see [MS-OXCMSG])
- PidNameKeywords (see [MS-OXCMSG])

The following properties are specific to the Exception Embedded Message object.

2.2.9.2.1 PidTagMessageClass

Type: **PtypString8**

The value of this property MUST be "IPM.OLE.class.{00061055-0000-0000-C000-000000000046}".

2.2.9.2.2 Best Body Properties

If the value of the PidLidFExceptionalBody property is FALSE, body properties SHOULD NOT be written to the Exception Embedded Message object. If body properties are written, then they follow the same rules that body properties for a Calendar object follow.

2.2.9.2.3 PidLidAppointmentStartWhole

Type: **PtypTime**

This property MUST exist on an Exception Embedded Message object, even if the exception has the same start date and time as the instance in the recurring series to which it corresponds. It contains the start date and time of the exception, and is specified in UTC.

2.2.9.2.4 PidLidAppointmentEndWhole

Type: **PtypTime**

This property MUST exist on an exception object, even if the exception has the same end date and time as the instance in the recurring series to which it corresponds. It contains the end date and time of the exception and is specified in UTC.

2.2.9.2.5 PidLidExceptionReplaceTime

Type: **PtypTime**

This property specifies the date and time within the recurrence pattern that the exception will replace. The value is specified in UTC. This property allows the Exception Attachment object to be found for a particular instance.

2.2.9.2.6 PidLidFExceptionalBody

Type: **PtypBoolean**

A value of TRUE for this property indicates that the Exception Embedded Message object has a body that differs from the Recurring Calendar object. If the value of this property is TRUE, the Exception Embedded Message object MUST have a body. If the value of this property is FALSE, or if the property does not exist, a client or server obtains the body from the Recurring Calendar object.

2.2.9.2.7 PidLidFInvited

Type: **PtypBoolean**

The value of this property for an Exception Embedded Message object takes the same meaning as specified in section 2.2.3.4. If a meeting request has been sent for an exception but not for the recurring series, the value of this property on the Recurring Calendar object will still be FALSE, but the value on the Exception Embedded Message object will be TRUE.

2.2.10 Calendar Folder

For a folder to be treated as a Calendar folder, it MUST have the properties specified in this section. When creating Calendar objects, the client or server SHOULD<51> create them in the **Calendar special folder**.

2.2.10.1 PidTagContainerClass

Type: **PtypString8**

The value of this property for all calendar folders MUST be set to "IPF.appointment.".

2.2.10.2 PidTagDefaultPostMessageClass

Type: **PtypString**

If this property is set on a calendar folder, the value MUST either contain "IPM.appointment", or begin with "IPM.appointment".

2.2.11 Delegate Information Object

The following properties are set on the **Delegate Information object**, as specified in [MS-OXODLGT].

2.2.11.1 PidTagFreeBusyCountMonths

Type: **PtypInteger32**

This property is used to calculate the start and end dates of the range of **Free/busy** data to be **published** to the **public folders** <52>, as specified in [MS-OXOPFFB]. The value of this property MUST be greater than or equal to 0x00000000 and less than or equal to 0x00000024. This is not a required property.

2.2.11.2 PidTagScheduleInfoAutoAcceptAppointments

Type: **PtypBoolean**

A value of TRUE for this property indicates that a client or server SHOULD automatically respond to all meeting requests for the attendee or resource. The response MUST be acceptance, unless an additional constraint specified by the PidTagScheduleInfoDisallowRecurringAppts or PidTagScheduleInfoDisallowOverlappingAppts property is met. A value of FALSE or the absence of this property indicates that a client or server does not automatically accept meeting requests. This is not a required property.

2.2.11.3 PidTagScheduleInfoDisallowRecurringAppts

Type: **PtypBoolean**

This property is only meaningful when the value of the PidTagScheduleInfoAutoAcceptAppointments property is TRUE. A value of TRUE indicates that when automatically responding to meeting requests, a client or server declines Meeting Request objects that represent a recurring series. A value of FALSE, or the absence of this property, indicates that recurring meetings are accepted. This is not a required property.

2.2.11.4 PidTagScheduleInfoDisallowOverlappingAppts

Type: **PtypBoolean**

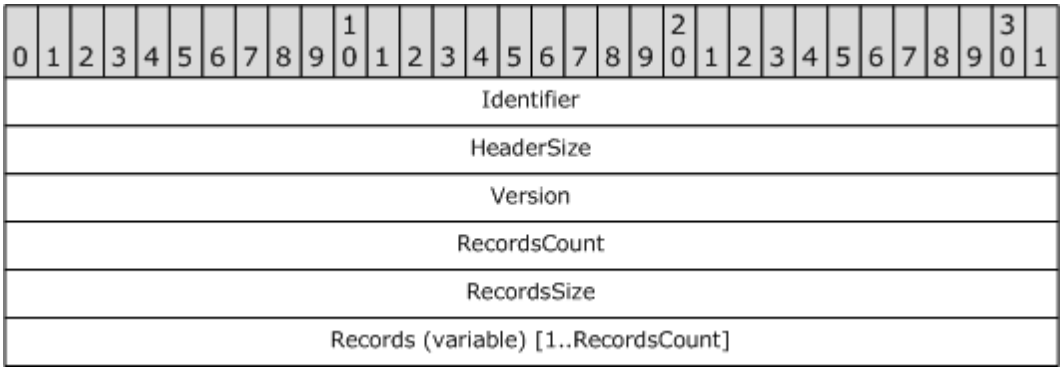
This property is only meaningful when the value of the PidTagScheduleInfoAutoAcceptAppointments property is TRUE. A value of TRUE indicates that when automatically responding to meeting requests, a client or server declines instances that overlap with previously scheduled events. A value of FALSE or the absence of this property indicates that overlapping instances are accepted. This is not a required property.

2.2.11.5 PidTagScheduleInfoAppointmentTombstone

Type: **PtypBinary**

This property in a delegator's Delegate Information object contains a list of tombstones. Each tombstone represents a Meeting object that has been declined. This is not a required property. If this property does not exist when a meeting is declined by the delegator or the delegate, it MUST be created.

This property has the following structure, where the fields are stored in little-endian byte order:



identifier: This field MUST have a value of 0xBEDEAFCD.

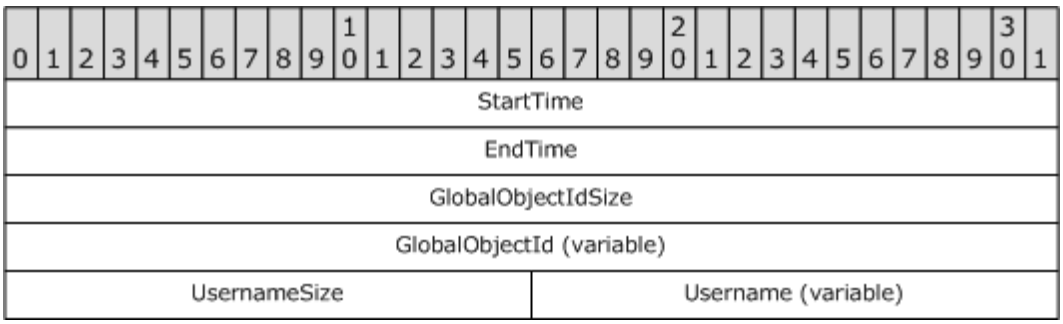
HeaderSize: This field MUST have a value of 0x00000014.

Version: This field MUST have a value of 0x00000003.

RecordsCount: The count of the Records field.

RecordsSize: This field MUST have a value of 0x00000014.

Records: An array of the Record data structure, where Record is defined as follows:



StartTime: The Meeting object's start time in minutes since midnight, January 1, 1601, UTC.

EndTime: The Meeting object's end time in minutes since midnight, January 1, 1601, UTC.

GlobalObjectIdSize: The size, in bytes, of the GlobalObjectId field.

GlobalObjectId: The value of the PidLidGlobalObjectId property of the meeting that this record represents.

UsernameSize: The size, in bytes, of the Username field.

Username: A non-Unicode string. The PidTagDisplayName of the Address Book object of the user who added the tombstone.

3 Protocol Details

There is no server role beyond those specified in [MS-OXCMSG] and [MS-OXOMSG].

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 specification does not mandate that implementations adhere to this model, as long as their external behavior is consistent with that described in this specification.

Objects specified in this document extend the Message object. The an abstract data model for these objects is the same as that specified in [MS-OXOMSG].

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Creating a Calendar Object

Although Appointment objects can be created in any Calendar folder, Meeting objects SHOULD only be created in the Calendar special folder (see [MS-OXOSFLD]). If a user creates a Meeting object in another Calendar folder, the client SHOULD<53> create a clone of the meeting in the Calendar special folder at the time of creation. All calendar objects MUST have all the required properties, as specified in section 2.2.1 and 2.2.2. A Meeting object MUST also have the required properties, as specified in section 2.2.3.

3.1.4.2 Converting an Appointment Object to a Meeting Object

To change an Appointment object into a Meeting object, the client sets the *asfMeeting* bit to 1 in the PidLidAppointmentStateFlags property. As long as a meeting request has not been sent for the Meeting object (according to the property PidLidFInvited), the client can set the *asfMeeting* bit to 0 (zero), reverting the Meeting object back to an Appointment object. However, after a meeting request is sent out, the *asfMeeting* bit MUST remain set to 1 on the Meeting object. In other words, the Meeting object MUST NOT revert to an Appointment object, even if all attendees are later removed.

3.1.4.3 Copying a Calendar Object

To copy a calendar object, the client creates a new calendar object in the target folder, and then copies all properties from the original object onto the new calendar object, with the exception of the following properties. <54>

The following properties MUST NOT be copied onto the new object:

- PidLidAppointmentColor
- PidLidGlobalObjectId
- PidLidCleanGlobalObjectId
- PidLidMeetingWorkspaceUrl

If the calendar object to be copied is a meeting object, the following actions MUST be taken by the client:

- The flag *auxApptFlagCopied* is added to the value of the PidLidAppointmentAuxiliaryFlags property on the new Object.
- The flag *asfReceived* SHOULD be added to the value of the PidLidAppointmentStateFlags property on the new Object<55>

In addition:

- The value of the PidLidFInvited property on the new object MUST be set to FALSE.
- The value of the PidTagOwnerAppointmentId property on the new object MUST be set to 0x00000000.
- The RecipientRows SHOULD be copied onto the new object. <56>
- The PidLidResponseStatus SHOULD<57> be set to respNotResponded.
- The PidTagSubjectPrefix property SHOULD<58> be set to to a localized string indicating the meeting is a copy.

3.1.4.3.1 Source Object is an Exception

When the source object is an exception, the client creates a new Calendar object. The client follows the same requirements for the new object, as already specified for copying a Calendar object. Furthermore, all properties that are not set on the Exception Embedded Message object but that are set on the Recurring Calendar object are to be copied onto the new object. In addition, the following actions MUST be taken by the client:

- The value of the PidTagMessageClass property MUST be reset to "IPM.appointment " on the new object.
- In addition to those already specified in section 3.1.4.3, the following properties MUST NOT be copied onto the new object:
  - PidLidAppointmentRecur
  - PidLidRecurrenceType
  - PidLidRecurrencePattern
  - PidLidTimeZoneStruct
  - PidLidTimeZoneDescription
  - PidLidFExceptionalAttendees

- The value of the PidLidClipStart property MUST be set to the value of the PidLidAppointmentStartWhole property.
- The value of the PidLidClipEnd property MUST be set to the value of the PidLidAppointmentEndWhole property.
- The value of the PidTagIconIndex property SHOULD be set to 0x00000400 if the Exception Attachment object was attached to an Appointment object or 0x00000402 if the Exception Attachment object was attached to a Meeting object.
- The value of the PidLidRecurring property MUST be set to FALSE.
- When copying the RecipientRows, the client copies them from the Exception Embedded Message object and not from the Recurring Calendar object.

3.1.4.3.2 Source is Not a Calendar Object

When the source object is not a Calendar object, the client creates a new Appointment object, and after copying all properties from the source object, ensure that all required properties (as specified in section 2.2.1 and 2.2.2) exist on the new Appointment object.

3.1.4.4 Deleting a Meeting Object

When the user deletes a Meeting object, the client SHOULD<59> send a Meeting Cancellation object to all attendees, as specified in section 3.1.4.8.1.

3.1.4.5 Recurrence Expansion

A client uses the **RecurrencePattern** structure specified in section 2.2.1.44.1 to enumerate the instances of the recurring series between the **StartDate** and **EndDate**. The client excludes every instance that occurs on a **DeletedInstanceDate** and includes every date in the **ModifiedInstanceDate** list. Note that **ModifiedInstanceDate** contains only the date on which the exception will occur and not its exact time. To get specific start and end dates and times for a given exception, the client uses the values from the **StartDateTime** and **EndDateTime** fields of the *ExceptionInfo* field specified in section 2.2.1.44.2.

3.1.4.5.1 Finding an Exception

The **AppointmentRecurrencePattern** structure specified in section 2.2.1.44.2 specifies deleted instances and modified instances. Every modified instance is associated with an Exception Attachment object, as specified in section 2.2.9. For each modified instance in the *RecurrencePattern*, there is a matching **ExceptionInfo** structure, as specified in section 2.2.1.44.2. The *StartDateTime* property is stored in the time zone represented by the PidLidTimeZoneStruct property that is stored on the Recurring Calendar object. To find the Exception Attachment object that corresponds to a modified instance, the *StartDateTime* field of the **ExceptionInfo** structure of that modified instance is matched to the PidLidAppointmentStartWhole property of the Exception Embedded Message object. The *StartDateTime* is converted to UTC by using PidLidTimeZoneStruct. This date and time SHOULD match the PidLidAppointmentStartWhole property of exactly one Exception Embedded Message object. If an Exception Attachment object cannot be found, the client creates a new one.

3.1.4.5.2 Creating an Exception

An exception replaces an instance of the recurring series. When creating a new exception, the client modifies the value of the PidLidAppointmentRecur property (as specified in section 2.2.1.44) in the

following way: The exception's new start date is added to the *ModifiedInstanceDate* array. *ModifiedInstanceCount* is incremented. The original start date is added to the *DeletedInstanceDate* array and the *DeletedInstanceCount* is incremented. The new and original start dates are in the time zone specified by PidLidTimeZoneStruct. The *ExceptionInfo*, as specified in section 2.2.1.44.2, is added to the recurrence BLOB. Note that the original start date and the new start date can be the same, if the date was not modified in the exception.

The client also adds an Exception Attachment object and Exception Embedded Message object, each with properties specified in section 2.2.9, and adds any overridden properties to the Exception Embedded Message object. The PidLidAppointmentStartWhole property of the Exception Embedded Message object is specified in UTC and is the UTC equivalent of the date and time added to StartDateTime in the *ExceptionInfo* field. The client also copies the *RecipientRows* from the Meeting object to the Exception Embedded Message object.

3.1.4.5.3 Deleting an Instance of a Recurring Series

To delete a single occurrence of a recurring series that is not a previously modified instance, the *DeletedInstanceCount* is incremented and the date of the instance being deleted is added to the *DeletedInstanceDate* array.

3.1.4.5.4 Deleting an Exception

To delete an exception, the instance being deleted is removed from the *ModificeInstanceDate* array and the *ModifiedInstanceCount* is decremented. The associated Exception Attachment object is also to be deleted.

3.1.4.6 Meeting Requests

3.1.4.6.1 Sending a Meeting Request

The organizer or delegate of the organizer sends a meeting request to inform attendees of the event. To do so, the client creates and submit a new Meeting Request object. The client copies all properties specified in section 2.2.1 from the Meeting object to the Meeting Request object. The client also adds all required properties specified in section 2.2.5. The client then sets the following on the Meeting Request object:

- The value of the PidLidAppointmentSequence property to zero.
- The *asfReceived* and *asfMeeting* bits on the PidLidAppointmentStateFlags property to 1.
- The value of the PidLidResponseStatus property to respNotResponded.
- The value of the PidLidIntendedBusyStatus property equal to the value of the PidLidBusyStatus property from the Meeting object.
- The value of the PidLidBusyStatus property to olTentative.
- The value of the PidLidFExceptionalAttendees property to FALSE.
- The value of the PidLidFExceptionalBody property to FALSE.
- The value of the PidLidIsRecurring property, as specified in section 2.2.1.13
- The value of the PidLidRecurring property, as specified in 2.2.1.12.

- The value of the PidLidCalendarType property, if the Meeting Request object represents a recurring series.
- The value of the PidLidWhere property equal to the value of the PidLidLocation property from the Meeting object.
- The value of the property PidLidAttendeeCriticalChange to the current date and time in UTC.
- The value of the PidLidMeetingType to mtgRequest.
- The value of the PidLidAllAttendeesString property, as specified in 2.2.1.16.
- The value of the PidLidToAttendeesString property, as specified in 2.2.1.17.
- The value of the PidLidCcAttendeesString property, as specified in 2.2.1.18.
- The value of the PidTagStartDate property, as specified in 2.2.1.30.
- The value of the PidTagEndDate property, as specified in 2.2.1.31.

The property PidTagProcessed is not set.

The following optional properties SHOULD also be set on the Meeting Request object:

- If the user has not modified the value of the PidLidReminderDelta property from its default value (as defined by the client), the value of this property SHOULD be set to the LONG value 0x5AE980E1.
- The client SHOULD prepend downlevel text to the body, as specified in 2.2.5.12.

After successfully sending a Meeting Request object, the client modifies the Meeting object in the organizer's Calendar folder in the following ways:

- Set the value of the PidLidFInvited property to TRUE.
- Set the value of the PidLidToAttendeesString property equal to the value that was set on the Meeting Request object.
- Set the value of the PidLidCcAttendeesString property equal to the value that was set on the Meeting Request object.

3.1.4.6.1.1 Direct Booking

The term "direct booking" refers to the action of creating a Meeting object directly on the Calendar folder of an attendee instead of sending a Meeting Request object to the attendee. A client MAY <60> attempt to direct book any sendable attendee as long as the following two **conditions** exist:

- The value of the PidTagScheduleInfoAutoAcceptAppointments property in the attendee's Delegate Information object is set to TRUE (see section 2.2.11.2).<61>
- The organizer has permission to write to the attendee's Calendar special folder (see [MS-OXCPERM]).

The client fails the direct booking action and does not send a Meeting Request object to any attendees if either of the following occurs:

- The value of the PidTagScheduleInfoDisallowRecurringAppts property in the attendee's Delegate Information object is set to TRUE and the Meeting Request object represents a recurring series (see section 2.2.11.2).
- The value of the PidTagScheduleInfoDisallowOverlappingAppts property (see section 2.2.11.2) in the attendee's Delegate Information object is set to TRUE and there is a meeting **conflict** during the date/time specified on the Meeting Request object. For details about how to determine whether a conflict exists, see section 3.1.4.10.

To direct book an attendee, the client takes the following actions:

- Create the Meeting object on the attendee's Calendar special folder, as specified in 3.1.4.6.2.2, and then modify the Meeting object as if the attendee had accepted it, as specified in 3.1.4.7.1. A Meeting Response object MUST NOT be sent to the organizer.
- publish updated free/busy information to the resource's Delegate Information object.
- Set the value of the PidTagRecipientTrackStatus property to respAccepted on the **RecipientRow** that represents the attendee on the organizer's Meeting object.
- Set the value of the PidTagRecipientTrackStatusTime property to the current date and time on the **RecipientRow** that represents the attendee in the organizer's Meeting object.
- If the Meeting Request object represents an exception, set the *recipExceptionalResponse* bit to 1 in the PidTagRecipientFlags property on the **RecipientRow** that represents the attendee in the organizer's Meeting object.
- Remove the RecipientRow that represents the attendee from the Meeting Request object so that it will not be sent to the attendee.

3.1.4.6.2 Receiving a Meeting Request

Some time after receiving a Meeting Request object, the client checks to determine whether the Calendar object is eligible for update, as specified in section 3.1.4.6.2.1, to determine whether to create a Meeting object in the user's Calendar special folder by using the information in the Meeting Request object. If the client does determine that the Meeting object has to be created, it creates the object as specified in section 3.1.4.6.2.2. If the *PiAutoProcess* value in the calendar Options dictionary [MS-OXOCFG] is set to 0 (zero), the client SHOULD NOT<62> immediately create the Meeting object, but wait until the user views the Meeting Request object. A client that does not support the calendar options dictionary can have its own defined mechanism for allowing the user to decide whether Meeting objects will be automatically created upon receipt of a Meeting Request object.

If the client decides to create the Meeting object, the client creates it according to the rules specified in section 3.1.4.6.2.1.

3.1.4.6.2.1 Deciding to Create a Meeting Object

When a delegator receives a Meeting Request object, the client follows the sequencing rules described in section 3.1.5.6 before deciding to automatically create a Meeting object.

If any one of the following conditions are met, the client does not automatically create the Meeting object:

- The Meeting Request object is located in the **Sent Mail folder** (see [MS-OXOSFLD]) or the Outbox **special Folder** (see [MS-OXOSFLD]).

- The value of the PidTagProcessed property on the Meeting Request object is set to TRUE.
- The Meeting Request object is intended for the delegator and a tombstone exists (as specified in section 2.2.11.5), indicating that another user has already declined the meeting.

The client MAY<63> skip automatic creation of the Meeting object if the value of the PidLidServerProcessed property on the Meeting Request object is set to TRUE, and the PidLidServerProcessingActions property either does not exist or has the *cpsCreatedOnPrincipal* bit of this property is set to 1. If the client skips automatic creation of the Meeting object, it MUST NOT set the PidTagProcessed property on the Meeting Request object.

3.1.4.6.2.2 Creating the Meeting Object

Before creating the Meeting object, the client tries to find the Calendar object, as specified in section 3.1.5.1, and does not create a new Meeting object if a match was found. After creating a Meeting object, the client copies all the properties specified in section 2.2.1 from the Meeting Request object onto the Meeting object. The client also adds all required properties specified in section 2.2.3. The client MAY<64> change the value of the PidTagMessageClass property on the new Meeting object to the value of the PidLidAppointmentMessageClass property from the Meeting Request object. In addition, the client sets the following properties on the Meeting object:

- The value of the PidLidResponseStatus property to respNotResponded.
- The value of the PidLidBusyStatus property to olTentative, unless the value of the PidLidIntendedBusyStatus property is olFree, in which case it MUST be set to olFree.
- If the value of the PidLidReminderDelta property in the Meeting Request object is set to 0x5AE980E1, change it to its default value (as defined by the client), and then recalculate the PidLidReminderSignalTime property, as specified in [MS-OXORMDR].
- The client SHOULD<65> copy the value of the PidLidAppointmentAuxiliaryFlags property from the Meeting Request object to the Meeting object.
- The client SHOULD remove the downlevel text (see 2.2.5.12) from the body.
- The client SHOULD<66> set the value of the PidLidAppointmentReplyName of the Meeting object to a null string.
- The client SHOULD<67> copy the **RecipientRows** in the PidLidAppointmentUnsendableRecipients property of the Meeting Request object to the RecipientRows of the Meeting object. For each **RecipientRow** copied, if the *recipOriginal* bit is set to 1 in the PidTagRecipientFlags property of the **RecipientRow**, then the client MUST set the *recipSendable* bit to 1 in the PidTagRecipientFlags property.
- The client MUST NOT copy the PidLidAppointmentUnsendableRecipients property from the Meeting Request object to the Meeting object.
- If the PidLidAppointmentUnsendableRecipients property is not set on the Meeting Request object, or if the client did not copy the **RecipientRows** in the PidLidAppointmentUnsendableRecipients property of the Meeting Request object to the Meeting object, then the client creates a **RecipientRow** for each recipient listed in the PidLidNonSendableTo, PidLidNonSendableCc, and PidLidNonSendableBcc properties. The client sets the recipient Type for each **RecipientRow** added as specified in section 2.2.1.19, 2.2.1.20, and 2.2.1.21
- The client sets the PidLidNonSendableTo, PidLidNonSendableCc, and PidLidNonSendableBcc properties to the null string on the Meeting object.

If the Meeting Request object represents a recurring series and the Meeting object was created, the client searches the folder for orphan instances of the meeting by matching the PidLidCleanGlobalObjectId property with that of the new Meeting object. The client then converts any orphan instances that are found into exceptions, and deletes the orphan instances. For each converted exception the client SHOULD<68> copy the PidLidBusyStatus from the orphan instance into the *BusyStatus* member of the associated *ExceptionInfo* field and set the ARO_BUSYSTATUS flag according to 2.2.1.44.2.

After creating the Meeting object, the client SHOULD set the value of the PidTagProcessed property on the Meeting Request object to TRUE, unless it is in a public folder, in which case this property is not set. <69>

After creating the Meeting object, the client MAY<70> set the PidLidServerProcessed property to TRUE. If setting the PidLidServerProcessed property, the client either sets both the *cpsCreatedOnPrincipal* and *cpsUpdatedCalItem* bits of the PidLidServerProcessingActions property to 1 or leaves this property unset. <71>

3.1.4.6.2.3 Auto Respond

After creating the Meeting object, the client can automatically send a Meeting Response object to the organizer if the value of the propertyPidTagScheduleInfoAutoAcceptAppointments in the organizer's Delegate Information object is nonzero. When sending the Meeting Response object, the client does so as specified in section 3.1.4.7. If the client chooses to automatically respond to Meeting Request objects, it also adheres to the requirements of the PidTagScheduleInfoDisallowRecurringAppts and PidTagScheduleInfoDisallowOverlappingAppts properties, accepting or declining meetings as appropriate.

The client MAY<72> skip automatic sending of Meeting Response objects to the organizer if the PidLidServerProcessed property of the Meeting Request object is set to TRUE and the *cpsSendAutoResponse* bit of the PidLidServerProcessingActions property is set to 1. If the client automatically responds to the Meeting Request object, it MAY<73> set the *cpsSendAutoResponse* bit of the PidLidServerProcessingActions property to 1.

When the client is acting for the delegate, and the client supports sending automatic responses, it uses the values defined for the delegator and not for the delegate when deciding whether or not to automatically respond to Meeting Request objects on behalf of the delegator.

3.1.4.6.3 Sending a Meeting Update

The organizer or delegate of the organizer sends an update to inform attendees of changes to an event that has already been sent out (according to the PidLidFInvited property on the Meeting object). To do so, the client creates and submits a Meeting Update object, following the same rules as sending a Meeting Request object (section 3.1.4.6.1), with differences as explained in this section.

If the value of the PidLidLocation property was modified by the user on the Meeting object, the client SHOULD set the value of the PidLidOldLocation property on the Meeting Update object to the old value. Similarly, if the value of the PidLidAppointmentStartWhole and/or PidLidAppointmentEndWhole properties were modified by the user on the Meeting object, the client SHOULD set the old values as the value of the PidLidOldWhenStartWhole and PidLidOldWhenEndWhole properties, respectively.<74>

The client modifies the sequence number as specified in section 3.1.5.4.

3.1.4.6.3.1 Significant Change

Certain constraints result when a "**significant change**" is made to a Meeting object. A significant change to a Meeting object includes any of the following conditions:

- The value of the propertyPidLidAppointmentStartWhole is changed.
- The value of the property PidLidAppointmentEndWhole is changed.
- The recurrence pattern, as defined in the property PidLidAppointmentRecur, was added, modified, or removed.

In the case that one of these significant changes has been made to the Meeting object, the value of the PidLidMeetingType property MUST be set to mtgFull. Otherwise, the value of this property SHOULD<75> be set to mtgInfo.

3.1.4.6.3.2 Clearing Previous Responses

If the Meeting object is set to request responses (according to the propertyPidTagResponseRequested), and a significant change (as specified in section 3.1.4.6.3.1) has been made, the client SHOULD clear all tallied responses that have been previously received from attendees. The client SHOULD NOT clear the tallied responses if a significant change has not been made, or if the Meeting object is not set to request responses.<76>

To clear the tallied responses, the client sets the value of the PidTagRecipientTrackStatus property to respNone in each **RecipientRow** of the Meeting object, as well as for any **RecipientRows** in the PidLidAppointmentUnsendableRecipients property and any recipients listed in the PidLidNonSendToTrackStatus, PidLidNonSendCcTrackStatus, and PidLidNonSendBccTrackStatus properties. The client also can set the value of the PidTagRecipientTrackStatusTime property in each RecipientRow to an invalid date<77>.

3.1.4.6.3.3 Adding Attendees to a Meeting

When the organizer adds a new attendee to a recurring series or single instance meeting, the client adds the attendee to the Meeting object's recipient rows and sets the properties as specified in section 2.2.3.9.

When the organizer adds a new attendee to an exception of a recurring series, the client adds a recipient row for the attendee to the Exception Embedded Message object. If the attendee already existed in the exception, but the *recipExceptionalDeleted* bit of the PidTagRecipientFlags property of the attendee's **RecipientRow** was set to 1, then the client resets this bit to 0.

3.1.4.6.3.4 Partial Attendee List

When a significant change (as specified in section 3.1.4.6.3.1) has not been made, and the user has added attendees, the client MAY <78> send the Meeting Update object to only the new attendees. The client SHOULD<79> treat an attendee as a new attendee if the value of the *recipSendable* bit of the attendee's PidTagRecipientFlags property has changed from 0 to 1. When sending a Meeting Update object to only new attendees, the client SHOULD<80> add all other attendees (for example, those not receiving the Meeting Update object) into the PidLidAppointmentUnsendableRecipients property on the Meeting Update object. For each attendee added to the PidLidAppointmentUnsendableRecipients property, the client sets the *recipOriginal* bit of the PidTagRecipientFlags property of the attendee's *RecipientRow* to 1 if the *recipSendable* bit is set to 1, and sets the *recipSendable* bit to 0.

3.1.4.6.3.5 Updating a Recurring Series

After a Meeting Update object is sent for a recurring series that has modified exceptions and the recurrence pattern has not changed, the client sends a Meeting Update object for each modified exception (according to the PidLidAppointmentStartWhole property on the Exception Embedded Message object) for which the start date and time has not yet passed. The Meeting Update object for each exception conforms to the specifications in section 2.2.5. Before sending a Meeting Update object for each exception, the client SHOULD<81> send a Meeting Cancellation object for that exception to each attendee included in the recurring series that is not included in the exception. If the attendee exists in the *RecipientRows* of the Exception object and the *recipExceptionalDeleted* bit of the PidTagRecipientFlags property of the attendee's *RecipientRow* is set to 1, then the client treats the attendee as not included in the exception. If the series has deleted exceptions, the client sends a Meeting Cancellation object for each deleted exception for which, (according to the *DeletedInstanceDates* field of the PidLidAppointmentRecur property of the Meeting object), the start date and time has not yet passed. The Meeting Cancellation object for each exception conforms to the specifications in section 2.2.5. If the recurrence pattern has changed, then the client SHOULD<82> send out Meeting Cancellation objects for each exception whose start date and time (according to the PidLidAppointmentStartWhole property on the Exception Embedded Message object) has not yet passed to every attendee of the exception, and removes every exception from PidLidAppointmentRecur and every Exception Attachment object.

After a Meeting Update object is sent to a Partial attendee List as defined in 3.1.4.6.3.3 for a recurring series that has exceptions, the client SHOULD<83> send a Meeting Request object for each exception whose start date and time (according to the PidLidAppointmentStartWhole property on the Exception Embedded Message object) has not yet passed to every attendee of the exception that is in the Partial attendee List.

3.1.4.6.4 Receiving a Meeting Update

Some time after receiving a Meeting Update object, the client determines whether to update the Meeting object in the user's Calendar special folder with the information in the Meeting Update object, as specified in section 3.1.4.6.4.1. If the client decides that the Meeting object has to be updated, it does so as specified in section 3.1.4.6.4.2. If the *PiAutoProcess* value in the calendar Options **dictionary** (see [MS-OXOCFG]) is set to 0 (zero), the client SHOULD NOT<84> immediately update the Meeting object, but will wait until the user views the Meeting Update object. A client that does not support the calendar Options dictionary can have its own defined mechanism for allowing the user to decide whether Meeting objects will be automatically updated upon receipt of a Meeting Update object.

3.1.4.6.4.1 Deciding to Update a Meeting Object

When a delegator receives a Meeting Update object, the client follows the sequencing rules described in section 3.1.5.6 before deciding to automatically update the Meeting object.

If any one of the following conditions is met, the client does not automatically update the Meeting object:

- The Meeting Request object is located in the Sent Mail folder or the Outbox special Folder (see [MS-OXOSFLD]).
- The value of the PidTagProcessed property on the Meeting Request object is set to TRUE.
- The Meeting Request object is intended for the delegator and a tombstone exists (as specified in 2.2.11.5), indicating that another user has already declined the meeting.

The client MAY<85> skip automatic updating of the Meeting object if the value of the PidLidServerProcessed property on the Meeting Request object is set to TRUE, and the PidLidServerProcessingActions property either does not exist or has the *cpsUpdatedCalItem* bit of this property is set to 1. If the client skips automatic updating of the Meeting object, it MUST NOT set the PidTagProcessed property on the Meeting Update object.

3.1.4.6.4.2 Updating the Meeting Object

When the client has determined that the Meeting object is eligible for an update, it first tries to find the Calendar object, as specified in section 3.1.5.1. If the Meeting Update object represents an exception, and the recurring series was found in the calendar, but the exception was previously deleted from the recurring series, then the client recreates the exception, as specified in section 3.1.4.5.2, unless the *cpsRevivedException* bit of the PidLidServerProcessingActions property of the Meeting Request object is set to 1 and the PidLidServerProcessed property is set to TRUE, in which case the client MAY <86> skip recreation of the exception. After recreating the exception, the client MAY <87> set the *cpsRevivedException* bit of the PidLidServerProcessingActions property of the Meeting Request object to TRUE. If the Meeting object was not found, then the client SHOULD change the value of the PidLidMeetingType property on the Meeting Update object to mtgRequest, and then follow the specification for receiving a new Meeting Request object, as specified in section 3.1.4.6.2.

If the user is the organizer of the meeting, then the client does not update the Calendar object with the information from the Meeting Update object.

If the Meeting Update object is out of date, as defined in section 3.1.5.2, the client SHOULD change the value of the PidLidMeetingType property on the Meeting Update object to mtgOutofDate and does not update the Meeting object. Similarly, if the Meeting Update object is not newer than the Meeting object, as defined in section 3.1.5.3, the client does not update the Meeting object.

Before modifying the Meeting object, the client SHOULD<88> do the following:

- Copy the value of the PidLidLocation property from the Meeting object onto the value of the PidLidOldLocation property on the Meeting Request object.
- Copy the value of the PidLidAppointmentStartWhole property from the Meeting object onto the value of the PidLidOldWhenStartWhole property on the Meeting Request object.
- Copy the value of the PidLidAppointmentEndWhole property from the Meeting object onto the value of the PidLidOldWhenEndWhole property on the Meeting Request object.
- The client MAY<89> skip these actions if the *cpsCopiedOldProperties* bit of the PidLidServerProcessingActions property of the Meeting Update object is set to 1 and the PidLidServerProcessed property is set to TRUE. The client MAY <90> set the *cpsCopiedOldProperties* bit of the PidLidServerProcessingActions property of the Meeting Update object to 1 after completing these actions.

To update the meeting, the client copies all the properties specified in section 2.2.1 from the Meeting Update object onto the Meeting object. The client also adds all required properties specified in section 2.2.3. However, the client SHOULD comply with the following exemptions:

- If the value of the PidTagSensitivity property (see [MS-OXCMSG]) on the Meeting object is set to private, it SHOULD<91> remain so, even if this is not the value of the property on the Meeting Update object.
- Remove the downlevel text (see section 2.2.5.12) from the body.

If the user had not yet responded to the original Meeting Request object, as reflected in the PidLidResponseStatus property on the Meeting object, the client has to ensure that the value of the PidLidMeetingType property on the Meeting Update object is mtgFull and the value of the PidTagIconIndex property on the Meeting Update object is 0x00000404.

If the Meeting Update object does not include a significant change (as specified in section 3.1.4.6.3.1), and the attendee had already responded to the original Meeting Request object, the client SHOULD NOT<92> change the value of the PidLidResponseStatus property on the Meeting object. Regardless of whether the attendee had previously responded, if the Meeting Update object represents an update with a significant change (as specified in section 3.1.4.6.3.1), the client sets the following properties on the Meeting object so that it looks as if the attendee has not yet responded:

- The value of the PidLidResponseStatus property to respNotResponded.
- The value of the PidLidBusyStatus property to olTentative, unless the value of the PidLidIntendedBusyStatus property is olFree, in which case it is set to olFree.

The client follows the same rules surrounding Auto Respond for a Meeting Update object, as specified for a Meeting Request object in section 3.1.4.6.2.3.

After updating the Meeting object, the client SHOULD set the value of the PidTagProcessed property to TRUE, unless the object is in a public folder, in which case this property is not to be set. <93>

After updating the Meeting object, the client MAY <94> set the PidLidServerProcessed property to TRUE. When setting the PidLidServerProcessed property, the client MUST either set the *cpsUpdatedCalItem* bit of the PidLidServerProcessingActions property to 1 or leave this property unset. <95>

3.1.4.6.5 Forwarding a Meeting Request

To forward a Meeting Request object, either from the organizer or from an attendee who received it, the client creates a new Meeting Request object and copies all the properties from the original Meeting Request object onto the new object. The client then makes the following additional changes on the new object:

- Set the value of the PidLidAttendeeCriticalChange property to the current date and time, in UTC.
- Set the value of the PidLidResponseStatus property to respNotResponded.
- Set the value of the PidLidBusyStatus property to olTentative, unless the value of the PidLidIntendedBusyStatus is olFree, in which case PidLidBusyStatus is set to olFree.
- Ensure that the asfMeeting and asfReceived bits are set to 1 in the PidLidAppointmentStateFlags property.
- Reset the value of the PidLidAllAttendeesString, PidLidToAttendeesString, and PidLidCcAttendeesString properties to a blank string.
- Set the value of the PidTagSenderName property to the value of the PidTagDisplayName property of the Address Book object of the forwarding user.
- Set the value of the PidTagSenderEntryId property to the value of the **EntryID** of the Address Book object of the forwarding user.
- Set the value of the PidTagSenderSearchKey property to the value of the SearchKey of the Address Book object of the forwarding user.

- Set the value of the PidTagSentRepresentingName property to the value of the PidTagDisplayName property of the Address Book object of the organizer.
- Set the value of the PidTagSentRepresentingEntryId property to the value of the EntryID of the Address Book object of the organizer.
- Set the value of the PidTagSentRepresentingSearchKey property to the value of the SearchKey of the Address Book object of the organizer.
- If the Meeting Request object represents an exception to a recurring series, set the value of the PidLidForwardInstance property to TRUE.
- Set the value of the PidLidChangeHighlight property to 0x00000000.
- Set the auxApptFlagForwarded bit to 1 in the PidLidAppointmentAuxiliaryFlags property.
- The client SHOULD<96> set the value of the PidLidMeetingType property to 0x00000001.

The client SHOULD copy all the RecipientRows from the original Meeting Request object into the PidLidAppointmentUnsendableRecipients <97> property of the new object. The client MUST NOT copy the RecipientRows from the original Meeting Request object into RecipientRows on the new object. The client can set the *auxApptFlagForceMtgResponse* bit in the PidLidAppointmentAuxiliaryFlags property. The property PidTagProcessed MUST NOT be set.

When a Meeting Request object is forwarded, the client MAY send a Meeting Forward Notification object to the organizer according to section 3.1.4.9.1.

3.1.4.6.5.1 Forwarding a Recurring Series

After a Meeting Request object is forwarded for a recurring series that has exceptions, the client SHOULD<98> forward each exception whose start date and time (according to the PidLidAppointmentStartWhole property on the Exception Embedded Message object) has not yet passed, as specified in 3.1.4.6.5.

3.1.4.7 Meeting Responses

3.1.4.7.1 Accepting a Meeting

When the attendee or a delegate of the attendee decides to accept a Meeting Request object, the client ensures that the Meeting object has been created in the attendee's Calendar special folder, as specified in section 3.1.4.6.2.2. Similarly, when the attendee or delegate of the attendee accepts a Meeting Update object, the client ensures that the Meeting object has been updated in the attendee's Calendar special folder, as specified in section 3.1.4.6.4.2, unless the Meeting Update object is out of date, as specified in section 3.1.5.2, in which case the client does not modify the Meeting object or send a Meeting Response object.

After creating or updating the Meeting object, all changes made to the Meeting object in the attendee's Calendar special folder MUST be atomic, for example, by creating a copy of the object, applying the changes to the copy, then deleting the original Meeting object. The client MUST make the following changes to the Meeting object:

- Set the value of the PidLidBusyStatus property equal to the value of the PidLidIntendedBusyStatus property from the Meeting Request object.
- Set the value of the PidLidResponseStatus property to respAccepted.

- Set the value of the PidLidAppointmentReplyTime property to the current date and time.
- If it is the delegate that is responding, set the value of the PidLidAppointmentReplyName property equal to the value of the PidTagMailboxOwnerName property from the store. If the delegate is not the one who is responding, the PidLidAppointmentReplyName property will not be not set.

The client MAY<99> send a Meeting Response object back to the organizer, as specified in 3.1.4.7.4.

3.1.4.7.2 Tentatively Accepting a Meeting

When the attendee or a delegate of the attendee decides to tentatively accept a Meeting Request object, the client follows the process specified in section 3.1.4.7.1, except that when updating the Meeting object, the following substitutions are made:

- Set the value of the PidLidBusyStatus property to olTentative, unless the value of the PidLidIntendedBusyStatus property is olFree, in which case it MUST be set to olFree.
- Set the value of the PidLidResponseStatus property to respTentative.

3.1.4.7.3 Declining a Meeting

When the attendee or a delegate of the attendee decides to decline a Meeting Request object, the client ensures that the Meeting object has been created in the attendee's Calendar special folder, as specified in section 3.1.4.6.2.2. Similarly, when the attendee or delegate of the attendee declines a Meeting Update object, the client has to ensure that the Meeting object has been updated in the attendee's Calendar special folder, as specified in section 3.1.4.6.4.2, unless the Meeting Update object is out of date, as specified in section 3.1.5.2, in which case the client MUST NOT modify the Meeting object and MUST NOT send a Meeting Response object.

After creating or updating the Meeting object, the client applies the following changes to the Meeting object in the attendee's Calendar special folder:

- If the value of the PidLidReminderSet property is set to TRUE, the Meeting object is not a recurring series, and the **signal time** has passed, set the value of the PidLidReminderSet property to FALSE.
- Set the value of the PidLidResponseStatus property to respDeclined.
- Set the value of the PidLidAppointmentReplyTime property to the current date and time.
- If the delegate is responding, set the value of the PidLidAppointmentReplyName property equal to the value of the PidTagMailboxOwnerName property from the store. If the delegate is not the one who is responding, the PidLidAppointmentReplyName property is not set.
- If it is the delegate acting on behalf of the delegator, the client SHOULD set the value of the PidLidOriginalStoreEntryId property to the EntryID of the delegator's store.

The following additional actions are performed by the client:

- If the Meeting Request object or Meeting Update object represents either a recurring series or single instance meeting, the client removes the Meeting object from the attendee's calendar, either by moving the Meeting object to the Deleted Items special Folder (see [MS-OXOSFLD]) or by permanently deleting the object.

- If the Meeting Request object or Meeting Update object represents an exception to a recurring series, the client removes the Exception Attachment object from the recurring series, as specified in section 3.1.4.5.4.
- If the delegator or a delegate acting on behalf of the delegator, declines a meeting, a tombstone SHOULD be added to the PidTagScheduleInfoAppointmentTombstone property on the delegator's Delegate Information object, as specified in section 3.1.5.6.

The client can send a Meeting Response object back to the organizer, as specified in 3.1.4.7.4.

3.1.4.7.4 Sending a Meeting Response

After choosing a response, an attendee or a delegate of the attendee sends a Meeting Response object to inform the organizer of the attendee's choice. The client SHOULD NOT send a Meeting Response object if one of the following conditions is true:

- The attendee is also the meeting organizer.<100>
- The value of the PidTagResponseRequested property on the Meeting Request object is set to FALSE<101>

If the following condition is true, the client can require sending a Meeting Response object to the organizer:

- The *auxApptFlagForceMtgResponse* bit is set to 1 in the value of the PidLidAppointmentAuxiliaryFlags property of the Meeting object (which came from the Meeting Request object or Meeting Update object).

Beyond these constraints, the client can send a Meeting Response object to the organizer to inform them of the attendee's choice. To do so, the client creates and submits a new Meeting Response object. The client then copies the following properties from the Meeting object to the Meeting Response object<102>:

- PidLidLocation
- PidLidWhere
- PidLidAppointmentSequence
- PidLidOwnerCriticalChange
- PidTagStartDate
- PidTagEndDate
- PidLidAppointmentStartWhole
- PidLidAppointmentEndWhole
- PidLidGlobalObjectId
- PidLidIsException
- PidTagOwnerAppointmentId
- PidTagSensitivity

In addition to these properties, if the Meeting Response object represents a recurring series, the client MUST copy the following properties from the Meeting object:

<103>

- PidLidTimeZoneStruct
- PidLidAppointmentRecur
- PidLidAppointmentTimeZoneDefinitionRecur
- PidLidIsRecurring
- PidLidTimeZone
- PidLidTimeZoneDescription

The client MUST also set the following on the Meeting Response object:

- The value of the PidTagMessageClass property as specified in section 2.2.6.1.
- The value of the PidTagIconIndex property as specified in section 2.2.1.49.
- The value of the PidLidAttendeeCriticalChange to the current date and time.
- The value of the PidTagSubjectPrefix property as specified in 2.2.6.2 to indicate the response type.
- Increment PidTagConversationIndex, as specified in [MS-OXOMSG].
- The value of the PidTagSentRepresentingName property to the value of the PidTagMailboxOwnerName property from the user's Mailbox (for example, a delegate acting on behalf of the delegator would write the name of the delegate).
- The value of the PidTagSentRepresentingEntryId property to the value of the PidTagMailboxOwnerEntryId property from the user's Mailbox.
- The value of the PidLidIsSilent property to TRUE if the user did not write any text in the body of the response.

3.1.4.7.4.1 Proposing a New Time

Along with the response, whether Accept, Tentatively Accept, or Decline, the attendee or a delegate of the attendee can request that the organizer change the meeting date and/or time. The client MUST NOT allow the attendee or delegate of the attendee to propose a new date or time in the following cases:

- The attendee is the organizer.
- The value of the PidLidAppointmentNotAllowPropose property on the Meeting Request object is set to TRUE.
- The Meeting Request object represents a recurring series. (However, the attendee can propose a new date and/or time for a single instance of a recurring series.)

To make the new date and/or time proposal, the client sets the following properties on the Meeting Response object :

- The value of the PidTagSubjectPrefix property as specified in section 2.2.6.2 to indicate a new date/time proposal.
- The value of the PidLidAppointmentCounterProposal property to TRUE.
- The value of the PidLidAppointmentProposedStartWhole property to the new proposed start date and time, in (UTC).
- The value of the PidLidAppointmentProposedEndWhole property to the new proposed end date and time, in UTC.
- The value of the PidLidAppointmentProposedDuration property to the new proposed duration, in minutes.

In addition to the previous information, when proposing a new date and/or time, the client MUST NOT set the value of the PidLidIsSilent property to TRUE, even if the attendee does not edit the body of the response.

3.1.4.7.5 Receiving a Meeting Response

Some time after receiving a Meeting Response object, the client decides, as specified in section 3.1.4.7.5.1, whether to record the attendee's response on the Meeting object in the organizer's Calendar special folder. If the client decides that the attendee's response needs to be recorded, it records the response as specified in section 3.1.4.7.5.2. If the PiAutoProcess value in the calendar Options dictionary (see [MS-OXOCFG]) is set to 0 (zero), the client SHOULD NOT<104> immediately record the response, but instead wait until the user views the Meeting Response object. A client that does not support the calendar Options dictionary can have its own defined mechanism for allowing the user to decide whether meeting responses will be automatically recorded upon receipt of a Meeting Response object.

3.1.4.7.5.1 Deciding to Record the Response

If any one of the following conditions is met, the client does not record the response for the attendee on the organizer's Meeting object:

- The Meeting Response object is located in the Sent Mail folder (see [MS-OXOSFLD]) or the Outbox special Folder (see [MS-OXOSFLD]).
- The value of the PidTagProcessed property on the Meeting Response object is set to TRUE.

The client SHOULD NOT<105> record the response for the attendee when the value of the PidLidServerProcessed property on the Meeting Response object is set to TRUE and the PidLidServerProcessingActions property either does not exist or has the cpsUpdatedCalItem bit of this property set to 1. <106>

3.1.4.7.5.2 Recording the Response

When the client has decided to record the response on the Meeting object, it finds the Calendar object, as specified in section 3.1.5.1. If the Meeting Response object represents an exception to a recurring series, and the recurring series was found in the calendar but it does not have an Exception Attachment object for this instance, one of two actions might need to be taken:

- If the instance was previously deleted from the recurring series on the organizer's meeting object, the client MUST NOT recreate the Exception Attachment object on the organizer's Meeting object just to record the response. Instead, the response is discarded.

- If the instance exists on the organizer's Meeting object but is not an exception, the Exception Attachment object is created on the organizer's Meeting object so that the response can be recorded.

If the Meeting Response object is found to be out of date, as specified in section 3.1.5.2, the client SHOULD<107> set the value of the PidLidPromptSendUpdate property on the Meeting Response object to TRUE and SHOULD<108> verify that a **RecipientRow** exists for the attendee, but the response MUST NOT be recorded.

To verify that a **RecipientRow** exists for the attendee, the client needs to find the **RecipientRow** that corresponds to the attendee in the organizer's Meeting object. If the client cannot find a **RecipientRow** for the attendee, it SHOULD<109> add a RecipientRow for the attendee as an optional attendee. If a **RecipientRow** for the attendee already exists, and the value of the PidTagRecipientTrackStatusTime property from the **RecipientRow** is a time that is later than the value of the PidLidAttendeeCriticalChange property on the Meeting Response object, the response from the Meeting Response object is not recorded. <110>

To record the response, the client sets the following on the RecipientRow:

- The value of the PidTagRecipientTrackStatus property to the appropriate value from the response table specified in section 2.2.1.11, according to the PidTagMessageClass property on the Meeting Response object.

PidTagMessageClass	PidTagRecipientTrackStatus
"IPM.Schedule.meeting.Resp.Pos"	respAccepted
"IPM.Schedule. meeting.Resp.Tent"	respTentative
"IPM.Schedule.meeting.Resp.Neg"	respDeclined

- The value of the PidTagRecipientTrackStatusTime property to the value of the PidLidAttendeeCriticalChange property from the Meeting Response object.<111>
- The *recipExceptionalResponse* bit to 1 in the PidTagRecipientFlags property, if the Meeting Response object represents an exception to a recurring series.

Regardless of whether the Meeting Response object includes a new date/time proposal, additional properties might need to be set. For more details about new date/time proposals, see section 3.1.4.7.5.3.

After recording the response, the client SHOULD<112> delete the Meeting Response object if the value of the PidLidIsSilent property is set to TRUE and the *piAutoDeleteReceipts* value in the calendar Options dictionary (see [MS-OXOCFG]) is set to True. A client that does not support the calendar Options dictionary MAY have its own defined mechanism for allowing the user to decide whether to automatically delete Meeting Response objects on which the PidLidIsSilent property is set to TRUE.

3.1.4.7.5.3 Handling New Date/Time Proposals

When the value of the PidLidAppointmentCounterProposal property on the Meeting Response object is set to TRUE, the attendee is proposing a new date and/or time. When this is the case, the client takes the following additional actions:

- Set the value of the PidTagRecipientProposed property to TRUE in the **RecipientRow** for the attendee.

- Set the value of the PidTagRecipientProposedStartTime property in the RecipientRow for the attendee equal to the value of the PidLidAppointmentProposedStartWhole property from the Meeting Response object.
- Set the value of the PidTagRecipientProposedEndTime property in the RecipientRow for the attendee equal to the value of the PidLidAppointmentProposedEndWhole property from the Meeting Response object.
- Set the value of the PidLidAppointmentCounterProposal property on the organizer's Meeting object to TRUE.
- If it is the first time this attendee has proposed a new date/time, increment the value of the PidLidAppointmentProposalNumber property on the organizer's Meeting object, by 0x00000001. If this property did not previously exist on the organizer's Meeting object, it MUST be set to the value of 0x00000001.

In light of the actions specified above, some actions might be required when a Meeting Response object is received without a new date/time proposal. Specifically, in the case where the attendee had previously proposed a new date/time (for example, the value of the PidTagRecipientProposed property in the RecipientRow for the attendee is already set to TRUE), and the new Meeting Response object does not have the property PidLidAppointmentCounterProposal set to TRUE, the client takes the following actions to undo the previous counter proposal :

- Set the value of the PidTagRecipientProposed property to FALSE in the **RecipientRow** for the attendee.
- Decrement the value of the PidLidAppointmentProposalNumber property on the organizer's Meeting object by 1.
- If the value of the PidLidAppointmentProposalNumber property becomes zero (meaning no other attendees have new date/time proposals), set the value of the PidLidAppointmentCounterProposal property on the organizer's Meeting object to FALSE.

3.1.4.8 Meeting Cancellations

3.1.4.8.1 Sending a Meeting Cancellation

The organizer or delegate of the organizer sends a Meeting Cancellation object to inform attendees that they no longer need to attend the event. To do so, the client creates and submits a new Meeting Cancellation object. The client then copies all properties from the Meeting object to the Meeting Cancellation object, with the exception/addition of those specified in section 1.3.1.6.

The client modifies the sequence number, as specified in section 3.1.5.4.

The client sets the following on the Meeting Cancellation object:

- All the bits in the value of the PidLidAppointmentStateFlags property that are set in this value on the Meeting object, and the *asfReceived* and *asfCanceled* bits to 1.
- The value of the PidLidResponseStatus property to respNotResponded.
- The value of the PidLidIntendedBusyStatus property to olFree.
- The value of the PidLidBusyStatus property to olFree.
- The value of the PidLidFExceptionalAttendees property to FALSE.

- The value of the PidLidFExceptionalBody property to FALSE.
- The property PidTagProcessed MUST NOT be set.
- The value of the PidTagSubjectPrefix property, as specified in section 2.2.7.1.

The following optional properties are also set on the Meeting Cancellation object:

- PidLidAllAttendeesString, as specified in section 2.2.1.16.
- PidLidToAttendeesString, as specified in section 2.2.1.17.
- PidLidCcAttendeesString, as specified in section 2.2.1.18.
- PidTagStartDate, as specified in section 2.2.1.30.
- PidTagEndDate, as specified in section 2.2.1.31.
- If the user has not modified the value of the PidLidReminderDelta property from its default value (as defined by the client), the value of this property SHOULD be set to the **LONG** value 0x5AE980E1.

After successfully sending a Meeting Cancellation object, the client does the following to modify the Meeting object in the organizer's Calendar folder :

- Set the value of the PidLidToAttendeesString property equal to the value that was set on the Meeting Cancellation object.
- Set the value of the PidLidCcAttendeesString property equal to the value that was set on the Meeting Cancellation object.

3.1.4.8.1.1 Partial Attendee List

When the organizer or delegate of the organizer removes attendees from the Meeting object, the client sends a Meeting Cancellation object to the attendees that were removed, but does not send a Meeting Cancellation object to any other attendees. If the organizer or delegate has changed the value of the *recipSendable* bit of the PidTagRecipientFlags property of any attendees from 1 to 0, the, the client SHOULD<113> send a cancelation to those attendees.

When sending a cancelation for a recurring series, the client removes the recipient rows corresponding to the attendees receiving cancelations from the Meeting object's recipient rows.

When sending a cancelation for an exception to a recurring series that is not a deleted exception, the client sets the *recipExceptionalDeleted* of the PidTagRecipientFlags property to 1 for each recipient row of the Exception Embedded Message object corresponding to the attendee receiving the cancelation.

3.1.4.8.1.2 Cancelling a Recurring Series

After a Meeting Cancellation object is sent to all attendees for a recurring series that has exceptions, the client sends a meeting Cancelation Object for each exception whose start date and time (according to the PidLidAppointmentStartWhole property on the Exception Embedded Message object) has not yet passed. The Meeting Cancellation object for each exception conforms to the specifications in section 2.2.7

If the series has deleted exceptions, the client SHOULD NOT<114> send a meeting Cancelation Object for each deleted exception the start date and time for which (according to the

DeletedInstanceDates field of the PidLidAppointmentRecur property of the Meeting object) has not yet passed.

After a Meeting Cancellation object is sent to a Partial attendee List as defined in 3.1.4.8.1.1, the client SHOULD<115>send a meeting Cancellation for each exception whose start date and time has not yet passed to every attendee of the exception that is also in the Partial attendee List. If sending a meeting Cancellation for an exception, the client sets the *recipExceptionalDeleted* bit of the PidTagRecipientFlags property to 1 for each removed attendee.

3.1.4.8.2 Receiving a Meeting Cancellation

Some time after receiving a Meeting Cancellation object, the client decides, as specified in section 3.1.4.8.2.1, whether to update the Meeting object in the user's Calendar special folder with the information in the Meeting Cancellation object. If the client decides that the Meeting object needs to be updated, it updates the object as specified in section 3.1.4.8.2.2. If the *PiAutoProcess* value in the calendar Options dictionary (see [MS-OXOCFG]) is set to 0 (zero), the client SHOULD NOT<116> immediately update the Meeting object, but wait until the user views the Meeting Cancellation object. A client that does not support the calendar Options dictionary can have its own defined mechanism for allowing the user to decide whether Meeting objects will be automatically updated upon receipt of a Meeting Cancellation object.

3.1.4.8.2.1 Deciding to Update a Meeting Object

When a delegator receives a Meeting Cancellation object, the client MUST follow the sequencing rules described in section 3.1.5.6 before deciding to automatically update the Meeting object.

If any one of the following conditions is met, the client does not automatically update the Meeting object:

- The Meeting Cancellation object is located in the Sent Mail folder (see [MS-OXOSFLD]) or the Outbox special Folder (see [MS-OXOSFLD]).
- The value of the PidTagProcessed property on the Meeting Cancellation object is set to TRUE.
- The client MAY<117>skip automatic updating of the Meeting object if the value of the PidLidServerProcessed property on the Meeting Cancellation object is set to TRUE and the PidLidServerProcessingActions property either does not exist or the cpsUpdatedCalItem bit of this property is set to 1. If the client skips automatic updating of the Meeting object, then it MUST NOT set the PidTagProcessed property on the Meeting Cancellation object.

As long as the client has decided to update the Meeting object, it first tries to find the Calendar object, as specified in 3.1.5.1. If the Meeting Update object represents an exception to a recurring series, and the recurring series was found in the calendar but the exception was previously deleted from the recurring series, the client SHOULD NOT<118> recreate the Exception Attachment object and the Exception Embedded Message object on the recurring Meeting object. If the Meeting object was not found at all, the client SHOULD NOT <119> recreate it.

If the Meeting Update object is out of date, as specified in section 3.1.5.2, the client SHOULD change the value of the PidLidMeetingType property on the Meeting Update object to mtgOutofDate but does not update the Meeting object. Similarly, if the Meeting Cancellation object is not newer than the Meeting object, as specified in section 3.1.5.3, the client does not update the Meeting object.

3.1.4.8.2.2 Updating the Meeting Object

To update the Meeting object, the client copies all the properties specified in 2.2.1 from the Meeting Update object onto the Meeting object.

After updating the Meeting object, the client SHOULD set the value of the PidTagProcessed property to TRUE, unless the object is in a public folder, in which case this property is not set. <120>

3.1.4.9 Meeting Forward Notifications

3.1.4.9.1 Sending a Meeting Forward Notification

When a Meeting Request object is forwarded (see section 3.1.4.6.5), the client can send a Meeting Forward Notification object to the organizer. The client does not send a Meeting Forward Notification object if one of the following conditions is true:

- The PidTagAddressType property of the organizer's Address Book object is not equal to "EX"
- The PidTagAddressType property of the organizer's Address Book object is equal to "EX", but the PidLidGlobalObjectId is of type ThirdPartyGlobalId, as specified in [MS-OXCICAL] section 2.2.1.20.26.
- The version number returned by the server in the results from EcDoConnectEx, as specified in [MS-OXCRPC], is greater than or equal to 8.0.0.0.

The client SHOULD NOT<121> send a Meeting Forward Notification object if the following condition is true:

- The asfReceived bit of the PidLidAppointmentStateFlags property of the corresponding Meeting object is not set.

To notify the organizer of the new attendees, the client creates and submits a new Meeting Forward Notification object. The client MUST copy the following properties from the Meeting object to the Meeting Forward Notification object<122>:

- PidNameSubject
- PidLidLocation
- PidLidWhere
- PidLidAppointmentSequence
- PidLidOwnerCriticalChange
- PidTagStartDate
- PidTagEndDate
- PidLidAppointmentStartWhole
- PidLidAppointmentEndWhole
- PidLidGlobalObjectId
- PidLidCleanGlobalObjectId
- PidLidIsException

- PidTagOwnerAppointmentId
- PidTagSensitivity
- PidTagResponseRequested

In addition to these properties, if the forwarded Meeting Request object represents a recurring series, the client copies the following properties from the Meeting object to the Meeting Forward Notification object:

<123>

- PidLidTimeZoneStruct
- PidLidAppointmentRecur
- PidLidAppointmentTimeZoneDefinitionRecur
- PidLidIsRecurring
- PidLidTimeZone
- PidLidTimeZoneDescription

The client MUST also set the following on the Meeting Forward Notification object:

- The value of the PidTagMessageClass property, as specified in section 2.2.8.1.
- The value of the PidTagIconIndex property, as specified in section 2.2.1.49.
- The value of the PidLidAttendeeCriticalChange to the current date and time.
- The value of the PidTagSubjectPrefix property, as specified in section 2.2.8.2
- Increment PidTagConversationIndex, as specified in [MS-OXOMSG].
- The value of the PidTagSentRepresentingName property to the value of the PidTagMailboxOwnerName property from the user's Mailbox (for example, a delegate acting on behalf of the delegator would write the name of the delegate).
- The value of the PidTagSentRepresentingEntryId property to the value of the PidTagMailboxOwnerEntryId property from the user's Mailbox.

In addition, the client copies each **RecipientRow** with the *recipSendable* bit set in the PidTagRecipientFlags property from the forwarded Meeting Request object's **RecipientRows** to the PidLidForwardNotificationRecipients property on the Meeting Forward Notification object.

3.1.4.9.2 Receiving a Meeting Forward Notification

Some time after receiving a Meeting Forward Notification object, the client decides, as specified in section 3.1.4.7.5.1, whether to add the attendees included in the Meeting Forward Notification object to the Meeting object. If the client decides to add the attendees to the Meeting object, it MUST do so as specified in section 3.1.4.9.2.2. If the PiAutoProcess value in the calendar Options dictionary (see [MS-OXOCFG]) is set to 0 (zero), then the client SHOULD NOT<124> immediately add the forwarded attendees to the Meeting object, but instead wait until the user views the Meeting Forward Notification object. A client that does not support the calendar Options dictionary can have its own defined mechanism for allowing the user to decide whether forwarded attendees will be copied to the Meeting object upon receipt of a Meeting Forward Notification object.

3.1.4.9.2.1 Deciding to Add the Forwarded Attendees to the Meeting Object

If any one of the following conditions is met, the client MUST NOT record the attendee on the organizer's Meeting object:

- The Meeting Forward Notification object is located in the Sent Mail folder (see [MS-OXOSFLD]) or the Outbox special Folder (see [MS-OXOSFLD]).
- The value of the PidTagProcessed property on the Meeting Forward Notification object is set to TRUE.
- The *cpsProcessedMeetingForwardNotification* bit of the PidLidServerProcessingActions property of the Meeting Forward Notification object is set to 1.

3.1.4.9.2.2 Adding the Forwarded Attendees to the Meeting Object

As long as the client has decided to add the forwarded attendees to the Meeting object, it MUST find the Calendar object, as specified in section 3.1.5.1. If the Meeting Forward Notification object represents an exception to a recurring series, and the recurring series was found in the calendar but it does not have an Exception Attachment object for this instance, one of two actions might need to be taken:

- If the instance was previously deleted from the recurring series on the organizer's meeting object, the client MUST NOT recreate the Exception Attachment object on the organizer's Meeting object just to add the attendee.
- If the instance exists on the organizer's Meeting object but is not an exception, the Exception Attachment object MUST be created on the organizer's Meeting object so that the response can be recorded.

To add the forwarded attendees to the Meeting object, the client MUST copy each **RecipientRow** in the PidLidForwardNotificationRecipients property of the Meeting Forward Notification object to the **RecipientRows** of the Meeting object if and only if the following conditions are met:

- The value of the **RecipientRow's** recipient Type is not 0x03.
- The recipient already exists in the Meeting object's RecipientRows according to the value of the PidTagEntryId property.

If the client copies a **RecipientRow** and the recipient Type of the **RecipientRow** is 0x01, the client MUST set the recipient Type of the corresponding **RecipientRow** on the Meeting object to 0x02.

If the Meeting Forward Notification object is out-of-date as specified in section 3.1.5.2, the client sets the value of the PidLidPromptSendUpdate property to TRUE.

After copying the forwarded attendees to the Meeting object, the client MUST set either the PidTagProcessed or PidLidServerProcessed property of the Meeting Forward Notification object to TRUE. If the client sets the PidLidServerProcessed property, the client MUST set the *cpsProcessedMeetingForwardNotification* bit of the PidLidServerProcessingActions of the Meeting Forward Notification object to 1.

3.1.4.10 Determining Meeting Conflicts

To determine whether a meeting conflicts with another meeting, follow these steps:

- Build a list of meetings that are in the range. Determine the range by using the start and end date/time of the given meeting as the start and end of the range. Any meeting the end date/time

for which is greater than or equal to the start date/time of the given meeting and the start date/time is less than or equal to the end date/time of the given meeting is considered to be in conflict.

- Expand any recurring meetings. For details about how to expand recurring meetings, see section 3.1.4.5. If multiple instances or exceptions fall into the range, each of them is considered as a single instance meeting for the purpose of this algorithm.

If the size of the list is greater than or equal to 1, the given meeting is considered to be in conflict.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Finding the Calendar Object

Several actions require finding the Calendar object to which a meeting-related object is referring. To find Calendar objects, the client searches in the Calendar special folder of the Mailbox that the event was intended for. This is typically the Mailbox of the user who is logged on, but for the delegate, the client searches the delegator's folder for objects received on behalf of the delegator.

To look for the object, the client first looks for a Calendar object for which the PidLidGlobalObjectId property matches the value of the PidLidCleanGlobalObjectId property of the meeting-related object.

If the action is being applied to an exception to a recurring series, the following additional operations are required, depending on whether a matching recurring series object was found:

- If a recurring series object was found, the client attempts to find the Exception Attachment object within a Calendar object by comparing the value of the PidLidExceptionReplaceTime property from the meeting-related object with either the PidTagExceptionReplaceTime property on the Exception Attachment object, or the PidLidExceptionReplaceTime property on the Exception Embedded Message object. Note that the PidTagExceptionReplaceTime property will not always be present on the Exception Attachment object. In the case where the Exception Attachment object cannot be found, a new one can be created.
- If the recurring series object was not found, the client looks for a recurring series object the PidLidGlobalObjectId property for which matches the value of the PidLidGlobalObjectId property of the meeting-related object. This would be the case, for example, if a user has been invited only to an exception to a recurring series.

3.1.5.2 Out-of-Date Meetings

A Meeting Request object or Meeting Update object becomes out of date when a more recent version is received and processed. A Meeting Response object is out of date when the attendee responds to an older Meeting Request object or Meeting Update object, instead of the most current Meeting Update object.

This section specifies how the client can determine whether the Meeting Request object or Meeting Response object is out of date. If one of the following conditions is true, the Meeting Request object or Meeting Response object is considered to be out of date:

- The value of the property PidLidMeetingType on the Meeting Request object is set to mtgOutofDate.
- The sequence number of the Meeting object is greater than that of the Meeting Request object or Meeting Response object.

- The sequence number of the Meeting object is the same as that of the Meeting Request object or Meeting Response object, but the value of the PidLidOwnerCriticalChange property on the Meeting Request object or Meeting Response object is earlier than the value of the "Request Time" property on the Meeting object, where "Request Time" is defined as follows:

recipient	Request Time
organizer	PidLidAppointmentSequenceTime
attendees	PidLidOwnerCriticalChange

- The value of the PidLidAttendeeCriticalChange property on the Meeting Response object is less than the value of the PidTagRecipientTrackStatusTime property on the **RecipientRow** of the organizer's Meeting object that represents the attendee.

3.1.5.3 Newer Meetings

A Meeting Request object or Meeting Cancellation object is considered to be from a newer version of the organizer's Meeting object than the Meeting object on the attendee's calendar if one of the following conditions is true:

- The sequence number on the Meeting Request object or Meeting Cancellation object is greater than the sequence number on the Meeting object.
- The sequence number on the Meeting Request object or Meeting Cancellation object equals the sequence number on the Meeting object, but the value of the PidLidOwnerCriticalChange property on the Meeting Request object or Meeting Cancellation object is greater than that of the Meeting object.

3.1.5.4 Incrementing the Sequence Number

When sending a Meeting Update object or Meeting Cancellation object, the client increments the sequence number except in the following cases:

- When sending a Meeting Cancellation object for a deleted exception after sending a Meeting Update object for a recurring series (See 3.1.4.6.5) the client does not increment the sequence number.
- When sending a Meeting Update object or Meeting Cancellation object for an exception to a recurring series, the client SHOULD<125> increment the sequence number.

If not incrementing the sequence number, the client sets the value of the PidLidAppointmentSequence property on the Meeting Update object or Meeting Cancellation object equal to the value of the PidLidAppointmentLastSequence property of the Meeting object

When incrementing the sequence number, the client sets the sequence number of the Meeting Update object or Meeting Cancellation object to a value greater than the sequence number that was set on any previous Meeting Request object, Meeting Cancellation object, or Meeting Update object. The client selects the greater of PidLidAppointmentLastSequence and PidLidAppointmentSequence properties from the Meeting object, and increment that value by 1, which results in the new sequence number. The client sets the new sequence number as the value of both the PidLidAppointmentLastSequence property on the Meeting object and the PidLidAppointmentSequence property on the Meeting Request object or the Meeting Cancellation object

If the Meeting Update object or Meeting Cancellation object is not being sent to all attendees of the meeting, then the client SHOULD NOT<126> set this new sequence number as the value of the PidLidAppointmentSequence property of the Meeting object.

3.1.5.5 Time Display Adjustments

In some cases, the client needs to make adjustments to the way in which it interprets PidLidAppointmentStartWhole and PidLidAppointmentEndWhole. Instead of interpreting these time properties as UTC values, a different process is followed for **floating appointments** (see section 3.1.5.5.1) and **time zone updates** (see 3.1.5.5.2).

3.1.5.5.1 Data Interpretation for Floating Appointments

The client SHOULD<127> interpret an object as a floating appointment if both of the following conditions are met:

- The value of PidLidAppointmentSubType property is TRUE
- The *asfMeeting* bit in the PidLidAppointmentStateFlags property is set to 0.

To correctly interpret the floating appointment the client MUST use the TZRule that is marked with the TZRULE_FLAG_EFFECTIVE_TZREG in the PidLidAppointmentTimeZoneDefinitionStartDisplay property to convert the values of the PidLidAppointmentStartWhole and PidLidAppointmentEndWhole properties from UTC to the time zone described by PidLidAppointmentTimeZoneDefinitionStartDisplay. The client MUST interpret these two time properties at this calculated time regardless of any additional time zone considerations. When performing these calculations, PidLidAppointmentTimeZoneDefinitionStartDisplay is used for all time properties, including PidLidAppointmentEndWhole.

3.1.5.5.2 Data Interpretation for Time Zone Updates

The **TZRule** that is marked with the **TZRULE_FLAG_EFFECTIVE_TZREG** in the property PidLidAppointmentTimeZoneDefinitionStartDisplay indicates the **TZRule** with which the Appointment object's times were converted to UTC time when the object was created. In some cases, the time zone rule that is in effect for the given time zone will be updated after the object is created.

When the client detects that the time zone rule for the time zone specified by PidLidAppointmentTimeZoneDefinitionStartDisplay has been updated, the client SHOULD<128> continue to interpret the PidLidAppointmentStartWhole and PidLidAppointmentEndWhole so that the values occur at the same time that was specified when the object was created. For example, if a user creates an Appointment object to begin at 2pm on April 1st in a time zone that has a -8 offset from UTC, PidLidAppointmentStartWhole would have been saved as 10pm UTC. If after creating this object, the time zone specified in PidLidAppointmentTimeZoneDefinitionStartDisplay is updated such that on April 1st the time zone's offset from UTC is now -7, the object's start time MUST continue to be interpreted as 2pm when PidLidAppointmentStartWhole is converted to that same time zone. The client can detect and perform these calculations using the data specified in PidLidAppointmentTimeZoneDefinitionStartDisplay. When performing these calculations, PidLidAppointmentTimeZoneDefinitionStartDisplay is to be used for all time properties, including PidLidAppointmentEndWhole.

If the object's times are being converted to a time zone that is different than the time zone specified by PidLidAppointmentTimeZoneDefinitionStartDisplay, the client MUST first convert PidLidAppointmentStartWhole and PidLidAppointmentEndWhole from UTC time to the time zone specified by the effective **TZRule**, and then use the updated time zone rule to convert to an updated UTC time prior to converting the time to another time zone.

3.1.5.6 Delegator Wants Copy

A value of TRUE for the PidTagScheduleInfoDelegatorWantsInfo property on the delegator's Delegate Information object indicates that the delegator only wants to be notified of meetings without taking action on them. When the delegator receives a Meeting Request object or Meeting Cancellation object, the client SHOULD<129> check the value of the PidTagScheduleInfoDelegatorWantsInfo property to see if it is set to TRUE unless one of the following holds true:

- The value of the PidLidMeetingType is mtgDelegatorCopy or mtgOutOfDate.
- The value of the PidLidServerProcessed property on the meeting-related object is TRUE and the value of the *cpsDelegatorWantsCopy* bit of the PidLidServerProcessingActions property on the meeting-related object is set to 1.
- The value of the PidTagSensitivity property (see [MS-OXCMSG]) is set to private.

If none of the above conditions holds true, and the client finds that the value of the PidTagScheduleInfoDelegatorWantsInfo property is TRUE, the client MUST change the value of the PidLidMeetingType property to mtgDelegatorCopy, and change the value of the PidTagIconIndex property to 0x00000409.

After checking whether or not the PidTagScheduleInfoDelegatorWantsInfo property is set to TRUE, the client MAY<130> set the *cpsDelegatorWantsCopy* bit of the PidLidServerProcessingActions property on the meeting-related object to 1.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

4.1 Examples of Properties

4.1.1 Recurrence BLOB Examples

Included in this section are several examples of the PidLidAppointmentRecur recurrence BLOB. The data for the fields of the **recurrence BLOB** are stored in little-endian byte ordering.

4.1.1.1 Recurrence BLOB Without Exceptions

The following example shows the binary recurrence data for an appointment that has the following characteristics:

- Occurs every Monday, Thursday, and Friday from 10:00 A.M. to 10:30 A.M.
- The recurrence ends after 12 occurrences.

The following is the recurrence binary large object (BLOB):

```
04300430 0B200100 0000C021 00000100 00000000 00000032 00000022 000000C0
00000000 00000000 00000000 00000000 00000008 020BC0C2 0ADBC0C0 63000000
93000058 02000076 02000000 00000000 00000000 00000000
```

The following table lists the content of the recurrence BLOB.

Name	Type	Size	Example	Description
ReaderVersion	WORD	2	04 30	This field indicates version 0x3004.
WriterVersion	WORD	2	04 30	This field indicates version 0x3004.
RecurFrequency	WORD	2	0b 20	The pattern of the recurrence is weekly.
PatternType	WORD	2	01 00	The pattern type is Week (0x0001).
CalendarType	WORD	2	00 00	The calendar type is Gregorian (0x0000).
FirstDateTime	ULONG	4	c0 21 00 00	1. Find the first **FirstDOW** before **StartDate**: 3/25/2007 2. Calculate the number of minutes between midnight that day and midnight, January 1, 1601: 213,654,240 3. Take that value modulo **Period**×10080 (The number of minutes in a week): 8640 (0x000021C0)

Name	Type	Size	Example	Description
Period	ULONG	4	01 00 00 00	The recurrence occurs every week (0x0001).
SlidingFlag	ULONG	4	00 00 00 00	The recurring instances do not rely on completion of the previous instances.
PatternTypeSpecific	Byte Array	Varies	32 00 00 00	The recurring appointment occurs on Monday, Thursday, and Friday. The value is determined by adding together the binary value of the decimal day mask (Sunday = 2^0 = 1, Monday = 2^1 = 2, Tuesday = 2^2 = 4, and so on). Monday (0x00000002) + Thursday (0x0000010) +Friday (0x00000020) = 0x00000032
EndType	ULONG	4	22 20 00 00	End after N occurrences. (0x00000222)
OccurrenceCount	ULONG	4	0C 00 00 00	The recurrence ends after 12 occurrences. 12 decimal value = 0x0C hexadecimal value.
FirstDOW	ULONG	4	00 00 00 00	The first day of the week on the calendar is Sunday (the default value).
DeletedInstanceCount	ULONG	4	00 00 00 00	There are no deleted instances.
ModifiedInstanceCount	ULONG	4	00 00 00 00	There are no modified instances.
StartDate	ULONG	4	80 20 BC 0C	The start date of the recurrence given in minutes since midnight January 1, 1601 corresponds to March 26, 2007 12:00:00 A.M.
EndDate	ULONG	4	20 AD BC 0C	The end date of the recurrence given in minutes since midnight January 1, 1601 corresponds to April 20, 2007 12:00:00 A.M.
ReaderVersion2	ULONG	4	06 30 00 00	This field indicates version 0x00003006.
WriterVersion2	ULONG	4	09 30 00 00	This field indicates version 0x00003009.
StartTimeOffset	ULONG	4	58 02 00 00	The hexadecimal start time of the recurrence is 0x00000258, which corresponds to 600 in decimal. 600 minutes is 10 hours, which is 10:00 A.M.
EndTimeOffset	ULONG	4	76 02 00 00	The hexadecimal end time of the recurrence is 0x000000276, which corresponds to 630 minutes, which is 10:30 A.M.
ExceptionCount	WORD	2	00 00	There are no exceptions in this recurrence BLOB.

Name	Type	Size	Example	Description
ReservedBlock1Size	ULONG	4	00 00 00 00	There is no data in the reserved block.
ReservedBlock2Size	ULONG	4	00 00 00 00	There is no data in the reserved block.

4.1.1.2 Weekly Recurrence BLOB with Exceptions

The following example shows the binary recurrence data for a meeting request.

The meeting request is the same as the request that is used in section 4.1.1.1, but in this example, the following information has been changed:

- Occurs every Monday, Thursday, and Friday from 10:00 A.M. to 10:30 A.M.
- The recurrence ends after 12 occurrences.
- The subject has been changed from 'Sample Recurrence' to 'Sample Recurrence with exception'.
- The location has been changed from 34/4639 to 34/4141.
- The start date and time has been modified from Monday 4/16/2007 10:00 A.M. to Monday 4/16/2007 11:00 A.M.
- The end date and time has been modified from Monday 4/16/2007 10:30 A.M. to Monday 4/16/2007 11:30 A.M.

The following is the recurrence BLOB:

```
04300430 0B200100 0000C021 00000100 00000000 00000032 00000022 000000C0
00000000 00000000 01000000 A096BC0C 01000000 A096BC0C 8020BC0C 20ADBC0C
06300000 09300000 58020000 76020000 01003499 BC0C5299 BC0CF898 BC0C1100
22002100 53696D70 6C652052 65637572 656E6365 20776974 68206578 63657074
696F6E73 00800700 33342F34 31343100 00000004 00000000 00000000 00000003
499BC0C5 299BC0CF 898BC0C2 10053006 9006D007 0006C006 50020005
20065006 30075007 20072006 5006E006 30065002 00077006 90074006 80020006
50078006 30065007 00074006 9006F006 E0073007 00730034 002F0034 00310034
00310000 00000000 00000000
```

Size: 0x0106 bytes

The following table lists the content of the modified recurrence BLOB.

Name	Type	Size	Example	Description
ReaderVersion	WORD	2	04 30	
WriterVersion	WORD	2	04 30	
RecurFrequency	WORD	2	0b 20	The pattern of the recurrence is weekly.
PatternType	WORD	2	01 00	The pattern type is Week (0x0001).
CalendarType	WORD	2	00 00	The calendar type is Gregorian (0x0000).
FirstDateTime	ULONG	4	c0 21 00 00	1. Find the first **FirstDOW** before **StartDate**:

Name	Type	Size	Example	Description
				3/25/2007 2. Calculate the number of minutes between midnight that day and midnight, January 1, 1601: 213,654,240 3. Take that value modulo **Period**×10080 (the number of minutes in a week): 8640 (0x000021C0)
Period	ULONG	4	01 00 00 00	The recurrence occurs every week (0x0001).
SlidingFlag	ULONG	4	00 00 00 00	The recurring instances do not rely on completion of the previous instances.
PatternTypeSpecific	Byte Array	Varies	32 00 00 00	The recurring appointment occurs on Monday, Thursday, and Friday. The value is determined by adding together the binary value of the decimal day mask (Sunday = 2^0 = 1, Monday = 2^1 = 2, Tuesday = 2^2 = 4, and so on). Monday (0x00000002) + Thursday (0x0000010) + Friday (0x00000020) = 0x00000032
EndType	ULONG	4	22 20 00 00	Ends after N occurrences. (0x00000222)
OccurrenceCount	ULONG	4	0C 00 00 00	The recurrence ends after 12 occurrences. 12 decimal value = 0x0C hexadecimal value.
FirstDOW	ULONG	4	00 00 00 00	The first day of the week on the calendar is Sunday (the default value).
DeletedInstanceCount	ULONG	4	01 00 00 00	There is one deleted instance.
DeletedInstanceDate	ULONG	4	A0 96 BC 0C	The date of the deleted instance is 4/16/2007 at 12:00:00 A.M.
ModifiedInstanceCount	ULONG	4	01 00 00 00	There is one modified instance.
ModifiedInstanceDate	ULONG	4	A0 96 BC 0C	The date of the modified or deleted instance is 4/16/2007 at 12:00:00 A.M.
StartDate	ULONG	4	80 20 BC 0C	The start date of the recurrence given in minutes since midnight January 1, 1601 corresponds to 3/26/2007 12:00:00 A.M.

Name	Type	Size	Example	Description
EndDate	ULONG	4	20 AD BC 0C	The end date of the recurrence given in minutes since midnight January 1, 1601 corresponds to 4/20/2007 12:00:00 A.M.
ReaderVersion2	ULONG	4	06 30 00 00	
WriterVersion2	ULONG	4	09 30 00 00	
StartTimeOffset	ULONG	4	58 02 00 00	The hexadecimal start time of the recurrence is 0x00000258, which corresponds to 600 in decimal. 600 minutes is 10 hours, which is 10:00 A.M.
EndTimeOffset	ULONG	4	76 02 00 00	The hexadecimal end time of the recurrence is 0x000000276, which corresponds to 630 minutes, which is 10:30 A.M.
ExceptionCount	WORD	2	01 00	One exception.
ExceptionInfo block				
StartDateTime	ULONG	4	34 99 BC 0C	The start date and time of the exception is 4/16/2007 at 11:00:00 A.M.
EndDateTime	ULONG	4	52 99 BC 0C	The end date and time of the exception is 4/16/2007 at 11:30:00 A.M.
OriginalStartTime	ULONG	4	F8 98 BC 0C	The original start date and time of the modified occurrence was 4/16/2007 at 10:00:00 A.M.
OverrideFlags	WORD	2	11 00	A value of 0x0011 indicates that two override flags are present: the ARO_SUBJECT (0x0001) and ARO_LOCATION (0x0010).
SubjectLength	WORD	2	22 00	The length of the subject including a null terminator is 34 characters.
SubjectLength2	WORD	2	21 00	The length of the subject is 33 characters.
Subject	Byte Array	Varies	53 69 6D 70 6C 65 20 52 65 63 75 72 72 65 6E 63 65 20 77 69 74 68 20 65 78 63 65	"Simple Recurrence with exceptions"

Name	Type	Size	Example	Description
			70 74 69 6F 6E 73	
LocationLength	WORD	2	08 00	The length of the location string including a null terminator is 8 characters.
LocationLength2	WORD	2	07 00	The length of the location string is 7 characters.
Location	Byte Array	Varies	33 34 2F 34 31 34 31	The modified location is "34/4141".
ReservedBlock1Size	ULONG	4	00 00 00 00	There is no data in this **skip block**.
ExtendedException block				
ChangeHighlight	Byte Array	Varies	04 00 00 00 00 00 00 00	The HighlightChange value is zero.
ReservedBlockEE1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
StartTime	ULONG	4	34 99 BC 0C	The start time of the recurrence is 4/16/2007 at 11:00:00 A.M.
EndTime	ULONG	4	52 99 BC 0C	The end time of the recurrence is 4/16/2007 at 11:30:00 A.M.
OriginalStartTime	ULONG	4	F8 98 BC 0C	The original start date and time of the recurrence was 4/16/2007 at 10:00:00 A.M.
WideCharSubjectLength	WORD	2	21 00	The length of the Unicode subject string is 33 characters.
WideCharSubject	Byte Array	Varies	53 00 69 00 6D 00 70 00 6C 00 65 00 20 00 52 00 65 00 63 00 75 00 72 00 72 00 65 00 6E 00 63	The modified Unicode subject is: "Simple recurrence with exceptions."

Name	Type	Size	Example	Description
			00 65 00 20 00 77 00 69 00 74 00 68 00 20 00 65 00 78 00 63 00 65 00 70 00 74 00 69 00 6F 00 6E 00 73 00	
WideCharLocationLength	WORD	2	07 00	The Unicode location string is 7 characters.
WideCharLocation	Byte Array	Varies	33 00 34 00 2F 00 34 00 31 00 34 00 31 00	The modified Unicode location is: "34/4141."
ReservedBlockEE2Size	ULONG	4	00 00 00 00	No data in this skip block.
ReservedBlock2Size	ULONG	4	00 00 00 00	No data in this skip block.

4.1.1.3 Daily Recurrence BLOB with Exceptions

The following example shows the binary recurrence data for an appointment that has the following characteristics:

- Occurs every 3 days, effective 4/7/2011 until 5/4/2011 from 8:00 A.M. to 8:30 A.M.
- The instances on 4/19/2011 and 4/22/2011 were deleted.

The following is the recurrence BLOB:

```
043004300A2000000000A0050000E0100000000000000212000000A0000000000000000020000000A0C1DC0C80D2DC0C0
0000000207EDC0C0016DD0C06300000093000000E0010000FE0100000000000000000000000000
```

Size: 0x0054 bytes

The following table lists the content of the modified recurrence BLOB.

Name	Type	Size	Example	Description
ReaderVersion	WORD	2	04 30	
WriterVersion	WORD	2	04 30	
RecurFrequency	WORD	2	0A 20	The pattern of the recurrence is daily.
PatternType	WORD	2	00 00	The pattern type is **Day** (0x0000).
CalendarType	WORD	2	00 00	The calendar type is Gregorian (0x0000).
FirstDateTime	ULONG	4	A0 05 00 00	For a daily recurrence, this value is numerical value of **StartDate** modulo **Period**.
Period	ULONG	4	E0 10 00 00	The recurrence occurs every 4320 minutes (3 days).
SlidingFlag	ULONG	4	00 00 00 00	The recurring instances do not rely on completion of the previous instances.
EndType	ULONG	4	21 20 00 00	Ends after an end date. (0x00002021)
OccurrenceCount	ULONG	4	0C 00 00 00	Not used.
FirstDOW	ULONG	4	00 00 00 00	The first day of the week on the calendar is Sunday (the default value).
DeletedInstanceCount	ULONG	4	02 00 00 00	There are two deleted instances.
DeletedInstanceDate	ULONG	4	A0 C1 DC 0C	The date of the deleted instance is 4/19/2007.
DeletedInstanceDate	ULONG	4	80 D2 DC 0C	The date of the deleted instance is 4/22/2007.
ModifiedInstanceCount	ULONG	4	00 00 00 00	There are no modified instances.
StartDate	ULONG	4	20 7E DC 0C	The start date of the recurrence is 4/7/2011.
EndDate	ULONG	4	00 16 DD 0C	The end date of the recurrence is 5/4/2011
ReaderVersion2	ULONG	4	06 30 00 00	
WriterVersion2	ULONG	4	09 30 00 00	
StartTimeOffset	ULONG	4	E0 01 00 00	The appointment's start time is 480 minutes past midnight or 8:00 AM.
EndTimeOffset	ULONG	4	FE 01 00 00	The appointment's end time is 510 minutes past midnight or 8:30 AM.

Name	Type	Size	Example	Description
ExceptionCount	WORD	2	00 00	No modified exceptions.
ReservedBlock1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
ReservedBlock2Size	ULONG	4	00 00 00 00	No data in this skip block.

4.1.1.4 N-Monthly Recurrence BLOB with Exceptions

The following example shows the binary recurrence data for an appointment that has the following characteristics:

- Occurs every third weekend day every 3 months starting at 2/9/2008 and ending after 10 occurrences.
- The instance on 5/10/2008 is moved to 5/11/2008.
- The location of the instance on 8/9/2008 is changed to "new location.".

The following is the recurrence BLOB for this recurrence:

```
043004300C200300000060AE000003000000000000041000000030000002220000000A000000000000000020000006
028C50C4028C70C020000000002EC50C4028C70C8028C30C6027D50C063000000930000048030000FC030000020048
31C50CFC31C50CA82BC50C0000882BC70C3C2CC70C882BC70C10000D000C006E6577206C6F6361746 96F6E0000000
004000000000000000000000000040000000000000000000000882BC70C3C2CC70C882BC70C0C006E006500770020000
6C006F00630061007400690 06F006E000000000000000000
```

Size: 0x00D2 bytes

The following table lists the content of the modified recurrence BLOB.

Name	Type	Size	Example	Description
ReaderVersion	WORD	2	04 30	
WriterVersion	WORD	2	04 30	
RecurFrequency	WORD	2	0C 20	The pattern of the recurrence is monthly.
PatternType	WORD	2	03 00	The pattern type is MonthNth (0x0003).
CalendarType	WORD	2	00 00	The calendar type is Gregorian (0x0000).
FirstDateTime	ULONG	4	60 AE 00 00	Find the first day of the month of the month of StartDate : 2/1/2008 Calculate the number of months between that midnight that day and midnight of the first day of the first month that falls in the Gregorian year

Name	Type	Size	Example	Description
				of 1601: 4885 Take that value modulo Period : 1 Add that number of months to the first day of the first month that falls in the Gregorian year 1601. 2/1/1601 Calculate the number of minutes between midnight that day and midnight, January 1, 1601. 44640 (0x0000AE60)
Period	ULONG	4	03 00 00 00	The recurrence occurs every 3 months.
SlidingFlag	ULONG	4	00 00 00 00	The recurring instances do not rely on the completion of the previous instances.
PatternTypeSpecific	Byte Array	Varies	41 00 00 00 03 00 00 00	The recurring appointment occurs on Saturday (0x00000040) and Sunday (0x00000001). The appointment occurs on the third occurrence of these days (0x00000003).
EndType	ULONG	4	22 20 00 00	End after N occurrences. (0x00000222).
OccurrenceCount	ULONG	4	0A 00 00 00	The recurrence ends after 10 occurrences.
FirstDOW	ULONG	4	00 00 00 00	The first day of the week on the calendar is Sunday (the default value).
DeletedInstanceCount	ULONG	4	02 00 00 00	There are two deleted instances.
DeletedInstanceDate	ULONG	4	60 28 C5 0C	The date of the deleted instance is 5/10/2008.
DeletedInstanceDate	ULONG	4	40 28 C7 0C	The date of the deleted instance is 8/9/2008.
ModifiedInstanceCount	ULONG	4	02 00 00 00	There are two modified instances.
ModifiedInstanceDate	ULONG	4	00 2E C5 0C	The date of the modified instance is 5/11/2008.
ModifiedInstanceDate	ULONG	4	40 28 C7 0C	The date of the modified instance is 8/9/2008.
StartDate	ULONG	4	80 28 C3 0C	The start date of the recurrence is 2/9/2008.

Name	Type	Size	Example	Description
EndDate	ULONG	4	60 27 D5 0C	The end date of the recurrence is 5/8/2010.
ReaderVersion2	ULONG	4	06 30 00 00	
WriterVersion2	ULONG	4	09 30 00 00	
StartTimeOffset	ULONG	4	48 03 00 00	The appointment's start time is 840 minutes past midnight, or 2:00 P.M.
EndTimeOffset	ULONG	4	FC 03 00 00	The appointment's end time is 1020 minutes past midnight, or 5:00 P.M.
ExceptionCount	WORD	2	02 00	Two exceptions.
ExceptionInfo block for exception 1:				
StartDateTime	ULONG	4	48 31 C5 0C	The start date and time of the exception is 5/11/2008 2:00 P.M.
EndDateTime	ULONG	4	FC 31 C5 0C	The end time of the exception is 5/11/2008 5:00 P.M.
OriginalStartTime	ULONG	4	A8 2B C5 0C	The original start date and time of the occurrence was 5/10/2008 2:00 P.M.
OverrideFlags	WORD	2	00 00	None.
ExceptionInfo block for exception 2:				
StartDateTime	ULONG	4	88 2B C7 0C	The start date and time of the exception is 8/9/2008 2:00 P.M.
EndDateTime	ULONG	4	3C 2C C7 0C	The end date and time of the exception is 8/9/2008 5:00 P.M.
OriginalStartTime	ULONG	4	88 2B C7 0C	The original start date and time of the occurrence was 8/9/2008 2:00 P.M.
OverrideFlags	WORD	2	10 00	ARO_LOCATION (0x00000010). The location has been modified.
LocationLength	WORD	2	0D 00	The length of the location string, including a null character, is 13.
LocationLength2	WORD	2	0C 00	The length of the location string is 12.
Location	Byte Array	Varies	6E 65 77 20 6C 6F 63 61 74 69 6F 6E	"new location"
ReservedBlock1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
ExtendedException block for exception 1:				

Name	Type	Size	Example	Description
ChangeHighlight	Byte Array	Varies	04 00 00 00 00 00 00 00	The size of the ChangeHighlight is 4. The value of the PidLidChangeHighlight property is zero for this exception.
ReservedBlockEE1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
ExtendedException block for exception 2:				
ChangeHighlight	Byte Array	Varies	04 00 00 00 00 00 00 00	The size of the ChangeHighlight is 4. The value of the PidLidChangeHighlight property is zero for this exception.
ReservedBlockEE1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
StartDateTime	ULONG	4	88 2B C7 0C	The start date and time of the exception is 8/9/2008 2:00 P.M.
EndDateTime	ULONG	4	3C 2C C7 0C	The end date and time of the exception is 8/9/2008 5:00 P.M.
OriginalStartTime	ULONG	4	88 2B C7 0C	The original start date and time of the occurrence was 8/9/2008 2:00 P.M.
WideCharLocationLength	WORD	2	0C 00	The length of the exception's Unicode location is 12 characters.
WideCharLocation	Byte Array	Varies	6E 00 65 00 77 00 20 00 6C 00 6F 00 63 00 61 00 74 00 69 00 6F 00 6E 00	"new location" in Unicode.
ReservedBlockEE2Size	ULONG	4	00 00 00 00	No data in this skip block.
ReservedBlock2Size	ULONG	4	00 00 00 00	No data in this skip block.

4.1.1.5 Yearly Recurrence BLOB with Exceptions

The following example shows the binary recurrence data for an appointment that has the following characteristics:

- Occurs every April 19, effective 4/19/2011 from 8:00 A.M. to 8:30 A.M.
- Move the instance on 4/19/2012 to 4/21/2012.

The following is the recurrence BLOB for this recurrence:

```
04300430D200200000040FA01000C0000000000000001300000023200000A000000000000000010000006 0CCE40C0
1000000A0D7E40CA0C1DC0CDF80E95A0630000009300000E0010000FE010000010080D9E40C9ED9E40C40CEE40C00
0000000000040000000000000000000000000000000000
```

Size: 0x0072 bytes

The following table lists the content of the modified recurrence BLOB.

Name	Type	Size	Example	Description
ReaderVersion	WORD	2	04 30	
WriterVersion	WORD	2	04 30	
RecurFrequency	WORD	2	0D 20	The pattern of the recurrence is yearly.
PatternType	WORD	2	02 00	The pattern type is Month (0x0002).
CalendarType	WORD	2	00 00	The calendar type is Gregorian.
FirstDateTime	ULONG	4	40 FA 01 00	1. Find the first day of the month of the month of StartDate: 4/1/2011 2. Calculate the number of months between midnight of that day and midnight of the first day of the first month that falls in the Gregorian year of 1601: 4/1/2011-1/1/1601 is 4887 months. 3. Take that value modulo Period: 4887 % 12 = 3. 4. Add that number of months to the first day of the first month that falls in the Gregorian year of the Gregorian year of 1601. 1/1/1601 + 3 months is 4/1/1601. 5. Calculate the number of minutes between midnight that day and midnight, January 1, 1601. 129,600 (0x0001FA40)
Period	ULONG	4	0C 00 00 00	The recurrence occurs every 12 months.
SlidingFlag	ULONG	4	00 00 00 00	The recurring instances do not rely on completion of the previous instances.
PatternTypeSpecific	Byte Array	Varies	13 00 00 00	The recurrence falls on the 19th of the month.
EndType	ULONG	4	23 20 00 00	Never ends. (0x00000232).
OccurrenceCount	ULONG	4	0A 00 00	Not used.

Name	Type	Size	Example	Description
			00	
FirstDOW	ULONG	4	00 00 00 00	The first day of the week on the calendar is Sunday (the default value).
DeletedInstanceCount	ULONG	4	01 00 00 00	There is one deleted instance.
DeletedInstanceDate	ULONG	4	60 CC E4 0C	The date of the deleted instance is 4/19/2012.
ModifiedInstanceCount	ULONG	4	01 00 00 00	There is one modified instance.
ModifiedInstanceDate	ULONG	4	A0 D7 E4 0C	The date of the modified instance is 4/21/2012.
StartDate	ULONG	4	A0 C1 DC 0C	The start date of the recurrence is 4/8/2008.
EndDate	ULONG	4	DF 80 E9 5A	The end date of the recurrence is never. (12/31/4500)
ReaderVersion2	ULONG	4	06 30 00 00	
WriterVersion2	ULONG	4	09 30 00 00	
StartTimeOffset	ULONG	4	E0 01 00 00	The appointment's start time is 480 minutes past midnight or 8:00 A.M.
EndTimeOffset	ULONG	4	FE 01 00 00	The appointment's end time is 510 minutes past midnight or 8:30 A.M.
ExceptionCount	WORD	2	01 00	One exception.
ExceptionInfo block for exception 1:				
StartDateTime	ULONG	4	80 D9 E4 0C	The start date and time of the exception is 4/21/2012 8:00 A.M.
EndDateTime	ULONG	4	9E D9 E4 0C	The end date and time of the exception is 4/21/2012 8:30 A.M.
OriginalStartTime	ULONG	4	40 CE E4 0C	The original start date and time of the occurrence was 4/19/2012 8:00 A.M.
OverrideFlags	WORD	2	00 00	None.
ReservedBlock1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
ExtendedException block for exception 1:				
ChangeHighlight	Byte Array	Varies	04 00 00 00 00 00 00	The size of the ChangeHighlight is 4. The value of the PidLidChangeHighlight property is zero for this exception.

Name	Type	Size	Example	Description
			00	
ReservedBlockEE1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
ReservedBlock2Size	ULONG	4	00 00 00 00	No data in this skip block.

4.1.1.6 Yearly Hebrew Lunar Recurrence BLOB with Exceptions

The following example shows the binary recurrence data for an appointment that has the following characteristics:

- Occurs every year on ג ו נ י ס starting ג' ו נ י ס ת ש ס"ח from 8:00 A.M. to 8:30 A.M.
- Change the Busy status of the second instance to ""Tentative"", make the reminder fire 60 minutes before the appointment, and change the body text.

The following is the recurrence BLOB for this recurrence:

```
04300430 0D200200 08008075 0A000C00 00000000 00000030 00000232 000000A0
00000000 00000000 10000002 07EDC0C0 10000002 07EDC0C6 074C40CD F80E95A0
63000000 93000000 E0010000 FE010000 01000080 DC0C1E80 DC0C0080 DC0C2402
3C000000 01000000 00000000 00040000 00000000 00000000 00000000 00000000
```

Size: 0x007A bytes

The following table contains the content of the modified recurrence BLOB.

Name	Type	Size	Example	Description
ReaderVersion	WORD	2	04 30	
WriterVersion	WORD	2	04 30	
RecurFrequency	WORD	2	0D 20	The pattern of the recurrence is yearly.
PatternType	WORD	2	02 00	The pattern type is Month (0x0002).
CalendarType	WORD	2	08 00	The calendar type is CAL_HEBREW (0x0008).
FirstDateTime	ULONG	4	0x000A7580	Find the first day of the month of the month of StartDate: 4/6/2008 (in Gregorian) Calculate the number of months between midnight of that day and midnight of the first day of the first month that falls in the Gregorian year of 1601: 4/6/2008-9/27/1601 is 4879 months. Take that value modulo Period: 4879 % 12 = 7 Add that number of months to the first day

Name	Type	Size	Example	Description
				of the first month that falls in the Gregorian year of the Gregorian year of 1601. 9/27/1601 + 7 Hebrew lunar months is 4/22/1602. Calculate the number of minutes between midnight of that day and midnight, January 1, 1601. 685,440 (0x000A7580)
Period	ULONG	4	0C 00 00 00	The recurrence occurs every 12 months.
SlidingFlag	ULONG	4	00 00 00 00	The recurring instances do not rely on completion of the previous instances.
PatternTypeSpecific	Byte Array	Varies	03 00 00 00	The recurrence falls on the third day of the month (in the Hebrew lunar calendar).
EndType	ULONG	4	23 20 00 00	Never ends. (0x00000232).
OccurrenceCount	ULONG	4	0A 00 00 00	Not used.
FirstDOW	ULONG	4	00 00 00 00	The first day of the week on the calendar is Sunday (the default value).
DeletedInstanceCount	ULONG	4	01 00 00 00	There is one deleted instance.
DeletedInstanceDate	ULONG	4	20 7E DC 0C	The date of the deleted instance is 4/7/2011.
ModifiedInstanceCount	ULONG	4	01 00 00 00	There is one modified instance.
ModifiedInstanceDate	ULONG	4	20 7E DC 0C	The date of the modified instance is 4/7/2011.
StartDate	ULONG	4	60 74 C4 0C	The start date of the recurrence is 4/8/2008.
EndDate	ULONG	4	DF 80 E9 5A	The end date of the recurrence is never. (12/31/4500)
ReaderVersion2	ULONG	4	06 30 00 00	
WriterVersion2	ULONG	4	09 30 00 00	
StartTimeOffset	ULONG	4	E0 01 00 00	The appointment's start time is 480 minutes past midnight or 8:00 A.M.
EndTimeOffset	ULONG	4	FE 01 00 00	The appointment's end time is 510 minutes past midnight or 8:30 A.M.
ExceptionCount	WORD	2	01 00	One exception.
ExceptionInfo block:				
StartDateTime	ULONG	4	00 80 DC 0C	The start date and time of the exception is 4/7/2011 8:00 A.M.

Name	Type	Size	Example	Description
EndDateTime	ULONG	4	1E 80 DC 0C	The end date and time of the exception is 4/7/2011 at 8:30 A.M.
OriginalStartTime	ULONG	4	00 80 DC 0C	The original start date and time of the occurrence was 4/7/2011 at 8:00 A.M.
OverrideFlags	WORD	2	24 02	A value of 0x0224 indicates that the following flags are set to 1 in this property: ARO_BUSYSTATUS | ARO_REMINDERDELTA | ARO_EXCEPTIONAL_BODY
ReminderDelta	ULONG	4	3C 00 00 00	The exception's value for PidLidReminderDelta is 60 (0x0000003C).
BusyStatus	ULONG	4	01 00 00 00	The exception's value for PidLidBusyStatus is 1.
ReservedBlock1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
ExtendedException block:				
ChangeHighlight	Byte Array	Varies	04 00 00 00 00 00 00 00	The size of the ChangeHighlight is 4. The value of the PidLidChangeHighlight property is zero for this exception.
ReservedBlockEE1Size	ULONG	4	00 00 00 00	There is no data in this skip block.
ReservedBlock2Size	ULONG	4	00 00 00 00	No data in this skip block.

4.1.2 Global Object ID Examples

This section includes examples of the PidLidGlobalObjectId and PidLidCleanGlobalObjectId BLOB properties that refer to an exception to a recurring series. The data for the fields of the Global Obj ID BLOB are stored in little-endian byte order, unless otherwise specified.

4.1.2.1 PidLidGlobalObjectId

The following is the value of the PidLidGlobalObjectId property for an object that represents an exception to a recurring series. The instance that is represented by the exception was moved from March 25, 2008 to March 26, 2008.

```
cb: 56
lpb:
040000008200E00074C5B7101A82E00807D803195025D461E473C801000000000000000100000002A5844B3A444F
74A9C246C60886F116B
```

Name	Type	Size	Sample	Description
Identifier	BYTE Array	16	04 00 00 00 82 00 E0 00 74 C5 B7 10	This byte array identifies the BLOB as a Global Object ID.

Name	Type	Size	Sample	Description
			1A 82 E0 08	
Year	WORD	2	07 D8	The original year of the instance represented by the exception. This value is in **big-endian** format instead of little-endian format. 0x07D8 (2008)
Month	BYTE	1	03	The original month of the instance represented by the exception. 0x03 (March)
Day	BYTE	1	19	The original day of the instance represented by the exception. 0x19 (25)
Creation Date	PtypTime	8	50 25 D4 61 E4 73 C8 01	2008/02/20 17:16:51
Reserved	Byte Array	8	00 00 00 00 00 00 00 00	
cbData	LONG	4	10 00 00 00	The length of the Data field. 0x00000010 (16) bytes
Data	Byte Array	16	2A 58 44 B3 A4 44 F7 4A 9C 24 6C 60 88 6F 11 6B	The data that uniquely identifies this Meeting object.

4.1.2.2 PidLidCleanGlobalObjectId

The following is the value of the PidLidCleanGlobalObjectId property for the exception from the example described in section 4.1.2.1. The only difference between these two properties is that in the clean version, the *Year*, *Month*, and *Day* fields are all 0 (zero).

```
cb: 56
lpb:
040000008200E00074C5B7101A82E008000000005025D461E473C80100000000000000000100000002A5844B3A444F
74A9C246C60886F116B
```

4.1.3 Downlevel Text for Meeting Request Body

A Meeting Request object can have extra body text with the date/time and location to help clients that do not understand the format, as specified in 2.2.5.12. The following is sample text from the body of a Meeting object:

```
Paulo,

This Friday I feel like eating out. How about we hit our old favorite?

- Jim
```

Figure 1:

The following shows how the body of the Meeting Request object might look to a client that does not understand the Meeting Request object format:

```
When: Thursday, February 28, 2008 12:00 PM-1:00 PM
Where: Our favorite restaurant

*~*~*~*~*~*~*~*~*~*

Paulo,

This Friday I feel like eating out. How about we hit our old favorite?

- Jim
```

Figure 2:

4.1.4 PidLidAppointmentTimeZoneDefinitionRecur BLOB

The following is an example of a PidLidAppointmentTimeZoneDefinitionRecur BLOB.

```
cb: 184 (0x000000B8)
lpb:
02013000020015005000610063006900660069006300200053007400610064006100720064002000540069006
D00650002002013E000000D607000000000000000000000000000000E00100000000000C4FFFFFF00000A00000005
000200000000000000000000400000001000200000000000000002013E000200D7070000000000000000000000000
0E00100000000000C4FFFFFF00000B00000001000200000000000000000300000002000200000000000000
```

The following table shows the content of this PidLidAppointmentTimeZoneDefinitionRecur BLOB.

Name	Type	Size	Example	Description
Major Version	BYTE	1	02	
Minor Version	BYTE	1	01	
cbHeader	WORD	2	30 00	**Header** contains 48 bytes.
TimeZoneDefinition flags	WORD	2	02 00	TZDEFINITION_FLAG_VALID_KEYNAME is set.
cchKeyName	WORD	2	15 00	KeyName has a length of 21 Unicode

Name	Type	Size	Example	Description
				characters.
KeyName	Unicode String, not terminated	Varies	50 00 61 00 63 00 69 00 66 00 69 00 63 00 20 00 53 00 74 00 61 00 6E 00 64 00 61 00 72 00 64 00 20 00 54 00 69 00 6D 00 65 00	"Pacific Time"
cRules	WORD	2	02 00	There will be two **TZRules**.
(Beginning of first **TZRule**)				
Major Version	BYTE	1	02	
Minor Version	BYTE	1	01	
Reserved	WORD	2	3E 00	
TZRule flags	WORD	2	00 00	This rule is not marked as the effective rule.
wYear	WORD	2	D6 07	This rule is applicable beginning January 1, 2006.
X	Byte Array	14	00 00 00 00 00 00 00 00 00 00 00 00 00 00	MUST be all zeros.
lBias	LONG	4	E0 01 00 00	This rule has a standard bias of 480 minutes from UTC.
lStandardBias	LONG	4	00 00 00 00	No additional bias during standard time.

Name	Type	Size	Example	Description
lDaylightBias	LONG	4	C4 FF FF FF	Daylight offset of -60 from the standard bias during daylight time.
stStandardDate	SYSTEMTIME	16	00 00 0A 00 00 00 05 00 02 00 00 00 00 00 00 00	This indicates the following **SYSTEMTIME** (in decimal): wYear: 0 wMonth: 10 wDayOfWeek: 0 wDay: 5 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 This means that the time zone will transition to standard time on the last Sunday of October at 2:00 A.M.
stDaylightDate	SYSTEMTIME	16	00 00 04 00 00 00 01 00 02 00 00 00 00 00 00 00	This indicates the following **SYSTEMTIME** (in decimal format): wYear: 0 wMonth: 4 wDayOfWeek: 0 wDay: 1 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 This means that the time zone will transition to daylight time on the first Sunday of April at 2:00 A.M.
(Beginning of second **TZRule**)				
Major Version	BYTE	1	02	
Minor Version	BYTE	1	01	
Reserved	WORD	2	3E 00	
TZRule flags	WORD	2	02 00	The TZRULE_FLAG_EFFECTIVE_TZREG flag is set to indicate that this rule is the effective rule.
wYear	WORD	2	D7 07	This rule is applicable beginning January 1, 2007.
X	Byte Array	14	00 00 00 00 00 00 00 00 00 00 00	MUST be all zeros.

Name	Type	Size	Example	Description
			00 00 00	
lBias	LONG	4	E0 01 00 00	This rule has a standard bias of 480 minutes from UTC.
lStandardBias	LONG	4	00 00 00 00	No additional offset during standard time.
lDaylightBias	LONG	4	C4 FF FF FF	Offset of -60 from the standard bias during daylight time.
stStandardDate	SYSTEMTIME	16	00 00 0B 00 00 00 01 00 02 00 00 00 00 00 00 00	This indicates the following **SYSTEMTIME** (in decimal): wYear: 0 wMonth: 11 wDayOfWeek: 0 wDay: 1 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 This means that the time zone will transition to standard time on the first Sunday of November at 2:00 A.M.
stDaylightDate	SYSTEMTIME	16	00 00 03 00 00 00 02 00 02 00 00 00 00 00 00 00	This indicates the following **SYSTEMTIME** (in decimal format): wYear: 0 wMonth: 3 wDayOfWeek: 0 wDay: 2 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 This means that the time zone will transition to daylight time on the second Sunday of March at 2:00 A.M.

4.1.5 PidLidTimeZoneStruct

The following is an example of a value for the PidLidTimeZoneStruct property.

```
cb: 48 (0x00000030)
lpb:
E00100000000000C4FFFFFF000000000B0000000100020000000000000000000000300000002000200000000000
000
```

The following table lists the content of the PidLidTimeZoneStruct BLOB.

Name	Type	Size	Example	Description
lBias	LONG	4	E0 01 00 00	This rule has a standard bias of 480 minutes from UTC.
lStandardBias	LONG	4	00 00 00 00	No additional offset during standard time.
lDaylightBias	LONG	4	C4 FF FF FF	Offset of -60 from the standard bias during daylight time.
wStandardYear	WORD	2	00 00	No year is specified, which indicates that the rule is a relative rule.
stStandardDate	SYSTEMTIME	16	00 00 0B 00 00 00 01 00 02 00 00 00 00 00 00 00	This indicates the following **SYSTEMTIME** (in decimal format): wYear: 0 wMonth: 11 wDayOfWeek: 0 wDay: 1 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 This means that the time zone will transition to standard time on the first Sunday of November at 2:00 A.M.
wDaylightYear	WORD	2	00 00	No year is specified, which indicates that the rule is a relative rule.
stDaylightDate	SYSTEMTIME	16	00 00 03 00 00 00 02 00 02 00 00 00 00 00 00 00	This indicates the following **SYSTEMTIME** (in decimal): wYear: 0 wMonth: 3 wDayOfWeek: 0 wDay: 2 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 This means that the time zone will transition to daylight time on the second Sunday of March at 2:00 A.M.

4.1.6 Sample of PidLidTimeZone

A PidLidTimeZone equal to 13 would indicate that the time zone has an offset from UTC+12 of 20*60 minutes, or 1200 minutes from UTC+12. This time zone has a daylight saving Standard Date of {11, 0, 1, 2}, equivalent to the first Sunday of November at 2:00 A.M. It has a Daylight Date of {3, 0, 2, 2}, equivalent to the second Sunday of March at 2:00 A.M.

4.2 Examples of Objects

Before manipulating an object, the client needs to ask the server to perform a mapping from **property names** to **property IDs**, using RopGetPropertyIdsFromNames. The following properties are referenced in the examples that follow.

property	**property set** GUID	**Name or ID**
PidLidAppointmentSequence	{ 00062002-0000-0000-c000-000000000046}	0x8201
PidLidAppointmentSequenceTime	{ 00062002-0000-0000-c000-000000000046}	0x8202
PidLidChangeHighlight	{ 00062002-0000-0000-c000-000000000046}	0x8204
PidLidBusyStatus	{ 00062002-0000-0000-c000-000000000046}	0x8205
PidLidAppointmentAuxiliaryFlags	{ 00062002-0000-0000-c000-000000000046}	0x8207
PidLidLocation	{ 00062002-0000-0000-c000-000000000046}	0x8208
PidLidAppointmentStartWhole	{ 00062002-0000-0000-c000-000000000046}	0x820D
PidLidAppointmentEndWhole	{ 00062002-0000-0000-c000-000000000046}	0x820E
PidLidAppointmentDuration	{ 00062002-0000-0000-c000-000000000046}	0x8213
PidLidAppointmentColor	{ 00062002-0000-0000-c000-000000000046}	0x8214
PidLidAppointmentSubType	{ 00062002-0000-0000-c000-000000000046}	0x8215
PidLidAppointmentRecur	{ 00062002-0000-0000-c000-000000000046}	0x8216
PidLidAppointmentStateFlags	{ 00062002-0000-0000-c000-000000000046}	0x8217
PidLidResponseStatus	{ 00062002-0000-0000-c000-000000000046}	0x8218
PidLidAppointmentReplyTime	{ 00062002-0000-0000-c000-000000000046}	0x8220
PidLidRecurring	{ 00062002-0000-0000-c000-000000000046}	0x8223
PidLidIntendedBusyStatus	{ 00062002-0000-0000-c000-000000000046}	0x8224

property	property set GUID	Name or ID
PidLidFInvited	{ 00062002-0000-0000-c000-000000000046}	0x8229
PidLidAppointmentReplyName	{ 00062002-0000-0000-c000-000000000046}	0x8230
PidLidRecurrenceType	{ 00062002-0000-0000-c000-000000000046}	0x8231
PidLidRecurrencePattern	{ 00062002-0000-0000-c000-000000000046}	0x8232
PidLidTimeZoneStruct	{ 00062002-0000-0000-c000-000000000046}	0x8233
PidLidTimeZoneDescription	{ 00062002-0000-0000-c000-000000000046}	0x8234
PidLidClipStart	{ 00062002-0000-0000-c000-000000000046}	0x8235
PidLidClipEnd	{ 00062002-0000-0000-c000-000000000046}	0x8236
PidLidAllAttendeesString	{ 00062002-0000-0000-c000-000000000046}	0x8238
PidLidAutoFillLocation	{ 00062002-0000-0000-c000-000000000046}	0x823A
PidLidToAttendeesString	{ 00062002-0000-0000-c000-000000000046}	0x823B
PidLidCcAttendeesString	{ 00062002-0000-0000-c000-000000000046}	0x823C
PidLidAppointmentNotAllowPropose	{ 00062002-0000-0000-c000-000000000046}	0x825A
PidLidAppointmentTimeZoneDefinitionStartDisplay	{ 00062002-0000-0000-c000-000000000046}	0x825E
PidLidAppointmentTimeZoneDefinitionEndDisplay	{ 00062002-0000-0000-c000-000000000046}	0x825F
PidLidAppointmentTimeZoneDefinitionRecur	{ 00062002-0000-0000-c000-000000000046}	0x8260
PidLidExceptionReplaceTime	{ 00062002-0000-0000-c000-000000000046}	0x8228
PidLidFExceptionalAttendees	{ 00062002-0000-0000-c000-000000000046}	0x822B
PidLidFExceptionalBody	{ 00062002-0000-0000-c000-000000000046}	0x8206

property	property set GUID	Name or ID
PidLidReminderDelta	{ 00062008-0000-0000-c000-000000000046}	0x8501
PidLidReminderTime	{ 00062008-0000-0000-c000-000000000046}	0x8502
PidLidReminderSet	{ 00062008-0000-0000-c000-000000000046}	0x8503
PidLidReminderSignalTime	{ 00062008-0000-0000-c000-000000000046}	0x8504
PidLidPrivate	{ 00062008-0000-0000-c000-000000000046}	0x8506
PidLidSideEffects	{ 00062008-0000-0000-c000-000000000046}	0x8510
PidLidCommonStart	{ 00062008-0000-0000-c000-000000000046}	0x8516
PidLidCommonEnd	{ 00062008-0000-0000-c000-000000000046}	0x8517
PidLidAttendeeCriticalChange	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0001
PidLidWhere	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0002
PidLidGlobalObjectId	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0003
PidLidIsSilent	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0004
PidLidIsRecurring	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0005
PidLidIsException	{6ed8da90-450b-101b-98da-00aa003f1305}	0x000A
PidLidTimeZone	{6ed8da90-450b-101b-98da-00aa003f1305}	0x000C
PidLidOwnerCriticalChange	{6ed8da90-450b-101b-98da-00aa003f1305}	0x001A
PidLidCalendarType	{6ed8da90-450b-101b-98da-00aa003f1305}	0x001C
PidLidCleanGlobalObjectId	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0023
PidLidAppointmentMessageClass	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0024

property	property set GUID	Name or ID
PidLidMeetingType	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0026
PidLidOldLocation	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0028
PidLidOldWhenEndWhole	{6ed8da90-450b-101b-98da-00aa003f1305}	0x0029
PidLidOldWhenStartWhole	{6ed8da90-450b-101b-98da-00aa003f1305}	0x002A

It is up to the server to keep track of, and return, the actual mapping. The following mapping values will be used in each of the examples in this section, as if the server had returned these values.

property	property ID
PidLidAppointmentSequence	0x81AF
PidLidAppointmentSequenceTime	0x82E7
PidLidChangeHighlight	0x82EC
PidLidBusyStatus	0x81B6
PidLidAppointmentAuxiliaryFlags	0x82D2
PidLidLocation	0x8009
PidLidAppointmentStartWhole	0x81B2
PidLidAppointmentEndWhole	0x81AC
PidLidAppointmentDuration	0x81A9
PidLidAppointmentColor	0x82CA
PidLidAppointmentSubType	0x8120
PidLidAppointmentRecur	0x81AD
PidLidAppointmentStateFlags	0x81B3
PidLidResponseStatus	0x8122
PidLidAppointmentReplyTime	0x8139
PidLidRecurring	0x81FD
PidLidIntendedBusyStatus	0x81E2
PidLidFInvited	0x81DA
PidLidAppointmentReplyName	0x81AE
PidLidRecurrenceType	0x81FE

property	property ID
PidLidRecurrencePattern	0x81FC
PidLidTimeZoneStruct	0x8214
PidLidTimeZoneDescription	0x8213
PidLidClipStart	0x81BA
PidLidClipEnd	0x81B9
PidLidAllAttendeesString	0x81A8
PidLidAutoFillLocation	0x82E8
PidLidToAttendeesString	0x82D9
PidLidCcAttendeesString	0x82DA
PidLidAppointmentNotAllowPropose	0x82D5
PidLidAppointmentTimeZoneDefinitionStartDisplay	0x83Aa8
PidLidAppointmentTimeZoneDefinitionEndDisplay	0x83A9
PidLidAppointmentTimeZoneDefinitionRecur	0x83AA
PidLidExceptionReplaceTime	0x83AC
PidLidFExceptionalAttendees	0x82D7
PidLidFExceptionalBody	0x82D8
PidLidReminderDelta	0x81FF
PidLidReminderTime	0x8005
PidLidReminderSet	0x8004
PidLidReminderSignalTime	0x8006
PidLidPrivate	0x82EF
PidLidSideEffects	0x8002
PidLidCommonStart	0x81BC
PidLidCommonEnd	0x81BB
PidLidAttendeeCriticalChange	0x81B5
PidLidWhere	0x8219
PidLidGlobalObjectId	0x81E0
PidLidIsSilent	0x81E6
PidLidIsRecurring	0x81E5
PidLidIsException	0x81E4

property	property ID
PidLidTimeZone	0x8212
PidLidOwnerCriticalChange	0x8128
PidLidCalendarType	0x81B7
PidLidCleanGlobalObjectId	0x81B8
PidLidAppointmentMessageClass	0x8311
PidLidMeetingType	0x8314
PidLidOldLocation	0x8316
PidLidOldWhenEndWhole	0x83CD
PidLidOldWhenStartWhole	0x83CC

4.2.1 Appointment Example

After making a dentist appointment for 10:00 A.M. (Pacific Daylight Time) on May 1, 2009, Mindy decides to set the information in her Calendar folder so that she will not forget about it. The appointment is an hour long, and she wants to be reminded about it half an hour before it happens. She wants to treat this as a private appointment, which indicates to a client to hide the details from other people. The following is a description of what a client might do to accomplish Mindy's intentions and the responses a server might return.

To create an Appointment object, the client uses RopCreateMessage. The server returns a success code and a handle to a Message object.

The client then uses RopSetProperties to transmit Mindy's data to the server.The following table shows an example of the data that might be sent by the client.

property	property ID	property type	Value
PidTagMessageClass	0x001a	0x001f (PtypString)	**IPM**.appointment
PidTagIconIndex	0x1080	0x0003 (PtypInteger32)	0x00000400
PidTagSensitivity	0x0036	0x0003 (PtypInteger32)	0x00000002 (SENSITIIVITY_PRIVATE)
PidLidPrivate	0x82ef	0x000b (PtypBoolean)	0x01 (TRUE)
PidLidSideEffects	0x8002	0x0003 (PtypInteger32)	0x00000171
PidLidCommonStart	0x81bc	0x0040 (PtypTime)	0x01c9ca7e43442800 (2009/05/01

property	property ID	property type	Value
			17:00:00.000)
PidLidCommonEnd	0x81bb	0x0040 (PtypTime)	0x01c9ca86a5089000 (2009/05/01 18:00:00.000)
PidLidReminderSet	0x8004	0x000b (PtypBoolean)	0x01 (TRUE)
PidLidReminderDelta	0x81ff	0x0003 (PtypInteger32)	0x0000001E (30)
PidLidReminderTime	0x8005	0x0040 (PtypTime)	0x01c9ca7e43442800 (2009/05/01 17:00:00.000)
PidLidReminderSignalTime	0x8006	0x0040 (PtypTime)	0x01c9ca7a1261f400 (2009/05/01 16:30:00.000)
PidLidBusyStatus	0x81b6	0x0003 (PtypInteger32)	0x00000002 (olBusy)
PidLidLocation	0x8009	0x001f (PtypString)	My Dentist's Office
PidLidAppointmentColor	0x82ca	0x0003 (PtypInteger32)	0x00000000
PidLidAppointmentStateFlags	0x81b3	0x0003 (PtypInteger32)	0x00000000
PidLidAppointmentAuxiliaryFlags	0x82d2	0x0003 (PtypInteger32)	0x00000000
PidLidAppointmentSubType	0x8120	0x000b (PtypBoolean)	0x00 (FALSE)
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000000 (respNone)
PidLidFInvited	0x81da	0x000b (PtypBoolean)	0x00 (FALSE)
PidLidAppointmentDuration	0x81a9	0x0003 (PtypInteger32)	0x0000003C (60)
PidLidAppointmentStartWhole	0x81b2	0x0040 (PtypTime)	0x01c9ca7e43442800 (2009/05/01

property	property ID	property type	Value
			17:00:00.000)
PidLidAppointmentEndWhole	0x81ac	0x0040 (PtypTime)	0x01c9ca86a5089000 (2009/05/01 18:00:00.000)
PidLidClipStart	0x81ba	0x0040 (PtypTime)	0x01c9ca7e43442800 (2009/05/01 17:00:00.000)
PidLidClipEnd	0x81b9	0x0040 (PtypTime)	0x01c9ca86a5089000 (2009/05/01 18:00:00.000)
PidLidRecurrenceType	0x81fe	0x0003 (PtypInteger32)	0x00000000
PidLidRecurring	0x81fd	0x000b (PtypBoolean)	0x00 (FALSE)
PidLidTimeZoneDescription	0x8213	0x001f (PtypString)	(GMT-08:00) Pacific Time (US & Canada)
PidLidAppointmentTimeZoneDefinitionStartDisplay	0x83a8	0x0102 (PtypBinary)	*1
PidLidAppointmentTimeZoneDefinitionEndDisplay	0x83a9	0x0102 (PtypBinary)	*1
PidLidGlobalObjectId	0x81e0	0x0102 (PtypBinary)	*2
PidLidCleanGlobalObjectId	0x81b8	0x0102 (PtypBinary)	*2

*1 = The start and end dates for this appointment are both set in the same time zone. See section 4.1.4 for an example of this **TimeZoneDefinition** BLOB. The time zone data for this appointment is as follows:

```
cb: 184
lpb:
02013000020015005000610063006900660069006300200053007400610072006400200054006900
6D00650020002013E000000D6070000000000000000000000000000E00100000000000C4FFFFFF00000A00000005
000200000000000000000004000000010002000000000000002013E000200D707000000000000000000000000000
0E00100000000000C4FFFFFF00000B00000001000200000000000000000300000002000200000000000000
```

*2 = This appointment is a single instance so the value of the PidLidGlobalObjectId and PidLidCleanGlobalObjectId properties are the same. See section 4.1.2 for an example ofthe Global Obj ID BLOB. The following is the value for this appointment:

```
cb: 56
```

```
lpb:
040000008200E00074C5B7101A82E0080000000020631F30F072C80100000000000000000010000000D97737CAB6762
A43BFF793851D08DB16
```

After setting all property values, the client can use RopSaveChangesMessage to commit the properties on the server. Without this, the newly created object will not be persisted. The server returns a success code that indicates that the data has been accepted.

Finally, the client uses RopRelease to release the handle that the server had returned from the initial RopCreateMessage.

4.2.2 Meeting Example

Mr. Glen John needs to set up a weekly half-hour meeting with a newly hired employee named Mr. Dennis Saylor. Mr. John likes to have meetings with team members on Tuesdays, and he is available at 10:30 A.M. The following sections provide a description of what a client might do to accomplish these tasks and the responses a server might return.

4.2.2.1 Creating the Meeting

To create the Meeting object, the client uses RopCreateMessage. The server returns a success code and a handle to a Message object.

The client then uses RopSetProperties to transmit Mr. John's data to the server. The following table shows an example of the data that might be sent by the client.

property	property ID	property type	Value
PidTagNormalizedSubject	0x0E1D	0x001F (PtypString)	Weekly meeting
PidTagSubjectPrefix	0x003D	0x001F (PtypString)	
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000002 (2)
PidLidAppointmentColor	0x82CA	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidLocation	0x8009	0x001F (PtypString)	Your Office
PidLidRecurring	0x81FD	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidAppointmentStartWhole	0x81B2	0x0040 (PtypTime)	0x01C878A5984A4400 (2008/02/26 18:30:00.000)
PidLidAppointmentEndWhole	0x81AC	0x0040 (PtypTime)	0x01C878A9C92C7800 (2008/02/26 19:00:00.000)

property	property ID	property type	Value
PidLidAppointmentDuration	0x81A9	0x0003 (PtypInteger32)	0x0000001E (30)
PidLidAppointmentAuxiliaryFlags	0x82D2	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidAppointmentSubType	0x8120	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidAppointmentStateFlags	0x81B3	0x0003 (PtypInteger32)	0x00000001 (1)
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000001 (respOrganized)
PidLidAppointmentNotAllowPropose	0x82D5	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidFInvited	0x81DA	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidRecurrenceType	0x81FE	0x0003 (PtypInteger32)	0x00000002 (2)
PidLidRecurrencePattern	0x81FC	0x001F (PtypString)	Every Tuesday from 10:30 A.M. to 11:00 A.M.
PidLidTimeZoneDescription	0x8213	0x001F (PtypString)	(GMT-08:00) Pacific Time (US & Canada)
PidLidClipStart	0x81BA	0x0040 (PtypTime)	0x01C8784D95BC0000 (2008/02/26 08:00:00.000)
PidLidClipEnd	0x81B9	0x0040 (PtypTime)	0x0CB2E57949B47A00 (4500/08/31 23:59:00.000)
PidLidToAttendeesString	0x82D9	0x001F (PtypString)	desaylor
PidLidAppointmentSequence	0x81AF	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidAutoFillLocation	0x82E8	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidReminderDelta	0x81FF	0x0003 (PtypInteger32)	0x0000000F (15)

property	property ID	property type	Value
PidLidReminderTime	0x8005	0x0040 (PtypTime)	0x01C878A5984A4400 (2008/02/26 18:30:00.000)
PidLidReminderSignalTime	0x8006	0x0040 (PtypTime)	0x01C878A37FD92A00 (2008/02/26 18:15:00.000)
PidLidCommonStart	0x81BC	0x0040 (PtypTime)	0x01C878A5984A4400 (2008/02/26 18:30:00.000)
PidLidCommonEnd	0x81BB	0x0040 (PtypTime)	0x01C878A9C92C7800 (2008/02/26 19:00:00.000)
PidLidReminderSet	0x8004	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidSideEffects	0x8002	0x0003 (PtypInteger32)	0x00000171 (369)
PidLidMeetingType	0x8314	0x0003 (PtypInteger32)	0x00000001 (1)
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.appointment
PidTagResponseRequested	0x0063	0x000B (PtypBoolean)	0x01 (TRUE)
PidTagIconIndex	0x1080	0x0003 (PtypInteger32)	0x00000403 (1027)
PidLidTimeZoneStruct	0x8214	0x0102 (PtypBinary)	*1
PidLidAppointmentTimeZoneDefinitionRecur	0x83AA	0x0102 (PtypBinary)	*2
PidLidAppointmentTimeZoneDefinitionStartDisplay	0x83A8	0x0102 (PtypBinary)	*3
PidLidAppointmentTimeZoneDefinitionEndDisplay	0x83A9	0x0102 (PtypBinary)	*3
PidLidGlobalObjectId	0x81E0	0x0102 (PtypBinary)	*4
PidLidCleanGlobalObjectId	0x81B8	0x0102 (PtypBinary)	*4
PidLidAppointmentRecur	0x81AD	0x0102 (PtypBinary)	*5

property	property ID	property type	Value
best body properties	A body **stream**, the text of which was written by Mr. John, that indicates to Mr. Saylor the purpose of the meeting. See [MS-OXBBODY] for details.		

*1 = See section 4.1.5 for an example of the PidLidTimeZoneStruct BLOB. The following is the value for this Meeting object:

```
cb: 48

lpb:
E00100000000000C4FFFFFF000000000B0000000100020000000000000000000030000000200020000000000
000
```

*2 = The PidLidAppointmentTimeZoneDefinitionRecur dates for this appointment are both set in the same time zone. See section 4.1.4 for an example of the **TimeZoneDefinition** BLOB. The only difference between this BLOB and that in PidLidAppointmentTimeZoneDefinitionStartDisplay / PidLidAppointmentTimeZoneDefinitionEndDisplay is that the TZRULE_FLAG_RECUR_CURRENT_TZREG flag is set in this BLOB. The following is the value for this Meeting object:

```
cb: 184

lpb:
02013000020015005000610063006900660069006300200053007400610 06E006400610072006400200054006900 6
D006500020002013E000000D607000000000000000000000000000000E00100000000000C4FFFFFF00000A00000005
000200000000000000000000400000001000200000000000000002013E000300D707000000000000000000000000000
0E00100000000000C4FFFFFF00000B000000010002000000000000000000030000000200020000000000000
```

*3 = The start and end dates for this appointment are both set in the same time zone. See section 4.1.4 for a an example of the **TimeZoneDefinition** BLOB. The following is the value for this Meeting object:

```
cb: 184
lpb:
02013000020015005000610063006900660069006300200053007400610 06E006400610072006400200054006900 6
D006500020002013E000000D607000000000000000000000000000000E00100000000000C4FFFFFF0 0000A00000005
000200000000000000000000400000001000200000000000000002013E000200D707000000000000000000000000000
0E00100000000000C4FFFFFF00000B000000010002000000000000000000030000000200020000000000000
```

*4 = This Meeting object is a recurring series, so the value of the PidLidGlobalObjectId and PidLidCleanGlobalObjectId properties are the same. See section 4.1.2 for a an example of the Global Obj ID BLOB. The following is the value for this Meeting object:

```
cb: 56
lpb:
040000008200E00074C5B7101A82E008000000000406FD661E473C801000000000000000100000002A5844B3A444F
74A9C246C60886F116B
```

*5 = Section 4.1.1.2 shows an example of the recurrence BLOB for a Weekly recurring meeting. The following is the value for this Meeting object:

```
cb: 80

lpb:
043004300B200100000C021000001000000000000000400000023200000A0000000000000000000000000000000002
088C30CDF80E95A0630000009300000760200009402000000000000000000000000
```

The client uses RopModifyRecipients to add Dennis Saylor to the Meeting object, including the extra properties listed in the following table.

property	property ID	property type	Value
PidTagRecipientFlags	0x5FFD	0x0003 (PtypInteger32)	0x00000201 (513)
PidTagRecipientTrackStatus	0x5FFF	0x0003 (PtypInteger32)	0x00000000 (0)

After setting all property values, the client can use RopSaveChangesMessage to commit the properties on the server. Without these properties, the newly created object will not be persisted. The server returns a success code that indicates that the data has been accepted.

4.2.2.2 Sending the Meeting Request

The client needs to use RopCreateMessage to create a new Meeting Request object in the Outbox special Folder so that attendees can be notified of the event. The server returns a success code and a handle to a new Message object.

Next, the client uses RopSetProperties to set on this new Meeting Request object all the properties that were set on the Meeting object as described in section 4.2.2.1, except for the following:

- PidLidBusyStatus
- PidLidAppointmentStateFlags
- PidLidResponseStatus
- PidLidFInvited
- PidLidAppointmentSequence
- PidLidAutoFillLocation
- PidLidReminderDelta*
- PidLidReminderSignalTime*
- PidLidSideEffects
- PidTagMessageClass
- PidTagIconIndex
- best body properties

* = The values of these **reminder properties** are not copied because the organizer kept the default reminder values. Instead, special values will be set on the Meeting Request object so that the receiving client uses default values that the attendee has defined.

In addition to the values that were already on the Meeting object, the client uses RopSetProperties to put the property values listed in the following table onto the Meeting Request object.

Property	Property ID	Property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.Schedule. meeting.Request
PidTagIconIndex	0x1080	0x0003 (PtypInteger32)	0xFFFFFFFF (-1)
PidTagStartDate	0x0060	0x0040 (PtypTime)	0x01C878A5984A4400 (2008/02/26 18:30:00.000)
PidTagEndDate	0x0061	0x0040 (PtypTime)	0x01C878A9C92C7800 (2008/02/26 19:00:00.000)
PidTagOwnerAppointmentId	0x0062	0x0003 (PtypInteger32)	0x4D9427D8 (1301555160)
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000001 (olTentative)
PidLidIntendedBusyStatus	0x81E2	0x0003 (PtypInteger32)	0x00000002 (olBusy)
PidLidAppointmentStateFlags	0x81B3	0x0003 (PtypInteger32)	0x00000003 (3)
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000005 (respNotResponded)
PidLidFInvited	0x81DA	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidAllAttendeesString	0x81A8	0x001F (PtypString)	desaylor
PidLidAppointmentSequence	0x81AF	0x0003 (PtypInteger32)	0x00000000 (0) If this had been an update, the sequence number would have been incremented.
PidLidChangeHighlight	0x82EC	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidReminderDelta	0x81FF	0x0003 (PtypInteger32)	0x5AE980E1 (1525252321)
PidLidReminderSignalTime	0x8006	0x0040 (PtypTime)	0x01C878A5984A4400 (2008/02/26 18:30:00.000)
PidLidSideEffects	0x8002	0x0003 (PtypInteger32)	0x00001C61 (7265)

Property	Property ID	Property type	Value
PidLidAttendeeCriticalChange	0x81B5	0x0040 (PtypTime)	0x01C874276FF4F450 (2008/02/21 01:16:51.093)
PidLidWhere	0x8219	0x001F (PtypString)	Your Office
PidLidAppointmentMessageClass	0x8311	0x001F (PtypString)	IPM.Appointment
PidLidIsRecurring	0x81E5	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidIsException	0x81E4	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidTimeZone	0x8212	0x0003 (PtypInteger32)	0x0000000D (13)
PidLidCalendarType	0x81B7	0x0003 (PtypInteger32)	0x00000001 (1)
PidLidOwnerCriticalChange	0x8128	0x0040 (PtypTime)	0x01C874276FF4F450 (2008/02/21 01:16:51.093)
best body properties	A body stream, the text of which is the downlevel text, as specified in section 2.2.5.12, followed by a copy of the body text from the Meeting object.		

In addition to these properties, the client needs to use RopSetProperties to add all properties that are required to send a Message object, as specified in [MS-OXOMSG], to the Meeting Request object so that it can be delivered to the attendee. This client also needs to use RopModifyRecipients to add a RecipientRow for Mr. Saylor to the Meeting Request object.

After the Meeting Request object has been created and filled in, it will be sent instead of saved. The client uses RopSubmitMessage to send this Message object for transport.

After the server returns a success code from submission, the client makes the changes listed in the following table to the Meeting object on Mr. John's calendar by using RopSetProperties.

Property	Property ID	Property type	Value
PidLidFInvited	0x81DA	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidAppointmentSequence	0x81AF	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidAppointmentSequenceTime	0x82E7	0x0040 (PtypTime)	0x01C874276FF4F450 (2008/02/21 01:16:51.093)
PidLidAttendeeCriticalChange	0x81B5	0x0040 (PtypTime)	0x0CB34557A3DD4000 (4501/01/01 00:00:00.000)
PidLidOwnerCriticalChange	0x8128	0x0040	0x01C874276FF4F450

Property	Property ID	Property type	Value
		(PtypTime)	(2008/02/21 01:16:51.093)
PidTagOwnerAppointmentId	0x0062	0x0003 (PtypInteger32)	0x4D9427D8 (1301555160)

Finally, the client issues RopSaveChangesMessage to save these changes to the organizer's Meeting object, and then releases both the meeting and Meeting Request objects by using a RopRelease for each.

4.2.2.3 Receiving the Meeting Request

After receiving the Meeting Request object, a client might tentatively add a Meeting object to the Calendar special folder in Mr. Saylor's Mailbox.

To accomplish this task, the client uses RopOpenMessage <131> to obtain a handle to the Meeting Request object, and RopCreateMessage to create a Meeting object in the Calendar special folder. The server returns a handle to each of these objects, along with appropriate success codes.

Next, the client uses RopSetProperties to set, on this new Meeting object, all the properties that were set on the Meeting Request object as described in 4.2.2.2, except for the following:

- PidTagMessageClass
- PidTagIconIndex
- PidLidChangeHighlight
- PidLidReminderDelta
- PidLidReminderSignalTime
- PidLidSideEffects
- best body properties

In addition to the values that were already on the Meeting object, the client uses RopSetProperties to put the property values listed in the following table onto the Meeting object.

property	property ID	property type	Value
PidLidReminderDelta	0x81FF	0x0003 (PtypInteger32)	0x0000000F (15) The default value for this client, given that the value on the Meeting Request object was 0x5AE980E1.
PidLidReminderSignalTime	0x8006	0x0040 (PtypTime)	0x01C878A37FD92A00 (2008/02/26 18:15:00.000)
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.appointment
PidTagIconIndex	0x1080	0x0003 (PtypInteger32)	0x00000403 (1027)

property	property ID	property type	Value
PidLidChangeHighlight	0x82EC	0x0003 (PtypInteger32)	0x00000E1F (3615)
PidLidSideEffects	0x8002	0x0003 (PtypInteger32)	0x00000171 (369)
best body properties	The client can look for and remove the downlevel text, as specified in section 2.2.5.12, before copying the text stream onto the new Meeting object.		

The client needs to set the recipients on the Meeting object by using RopModifyRecipients. The recipients are obtained from the RecipientRows of the Meeting Request object, as well as the PidLidAppointmentUnsendableRecipients property. In addition, if the organizer (in this case, Mr. John) is not in the list of recipients, his information is obtained from the PidTagSentRepresenting * properties and added as a RecipientRow.

After setting all property values, the client can use RopSaveChangesMessage to commit the properties on the server. Without this, the newly created object will not be persisted. The server returns a success code that indicates that the data has been accepted.

The client sets the following property on the Meeting Request object by using RopSetProperties, followed by RopSaveChangesMessage.

property	property ID	property type	Value
PidTagProcessed	0x7D01	0x000B (PtypBoolean)	0x01 (TRUE)

Finally, the client uses RopRelease to release the handle of the Meeting object and Meeting Request object.

4.2.2.4 Accepting the Meeting Request

After receiving the Meeting Request object that was, Mr. Dennis Saylor decides he will attend the meeting with Mr. Glen John. The client needs to send a Meeting Response object back to Mr. John so that he knows that Mr. Saylor will be in attendance.

To accomplish this task, the client uses RopOpenMessage to obtain a handle to the Tentative Meeting object, and RopCreateMessage to create a Meeting object in the Calendar special folder. The server returns a handle to each of these objects, along with appropriate success codes.

The client uses RopCopyTo to copy all properties from the Tentative Meeting object to the new Meeting object. The properties listed in the following table are then modified on the new Meeting object by using RopSetProperties.

property	property ID	property type	Value
PidLidAppointmentMessageClass	0x8311	0x001F (PtypString)	IPM.appointment
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000002 (olBusy)
PidLidResponseStatus	0x8122	0x0003 (	0x00000003 (respAccepted)

property	property ID	property type	Value
		PtypInteger32)	
PidLidAppointmentReplyTime	0x8139	0x0040 (PtypTime)	0x01C87427BCCA9A00 (2008/02/21 01:19:00.000)
PidLidAppointmentReplyName	0x81AE	0x001F (PtypString)	desaylor

The client uses RopSaveChangesMessage to persist the new Meeting object in Mr. Saylor's Calendar special folder. It releases a handle to the Tentative Meeting object by using RopRelease, and then deletes the Tentative Meeting object by using RopDeleteMessages.

Now the client needs to respond to the organizer. It uses RopCreateMessage to create a new Meeting Response object in the Outbox special Folder. The server returns a success code and a handle to a new Message object.

The client uses RopGetPropertiesSpecific on the Meeting object and then uses RopSetProperties to copy, onto this new Meeting Response object, the value of the following properties that were on the Meeting object:

- PidTagNormalizedSubject
- PidLidBusyStatus
- PidLidAppointmentColor
- PidLidLocation
- PidLidRecurring
- PidLidAppointmentStartWhole
- PidLidAppointmentEndWhole
- PidLidAppointmentTimeZoneDefinitionStartDisplay
- PidLidAppointmentTimeZoneDefinitionEndDisplay
- PidLidAppointmentDuration
- PidLidAppointmentAuxiliaryFlags
- PidLidAppointmentSubType
- PidLidAppointmentRecur
- PidLidRecurrenceType
- PidLidRecurrencePattern
- PidLidTimeZoneStruct
- PidLidAppointmentTimeZoneDefinitionRecur
- PidLidTimeZoneDescription

- PidLidClipStart
- PidLidClipEnd
- PidLidAppointmentSequence
- PidLidCommonStart
- PidLidCommonEnd
- PidLidWhere
- PidLidGlobalObjectId
- PidLidCleanGlobalObjectId
- PidLidAppointmentMessageClass
- PidLidIsRecurring
- PidLidIsException
- PidLidTimeZone
- PidLidCalendarType
- PidLidOwnerCriticalChange
- PidTagStartDate
- PidTagEndDate
- PidTagOwnerAppointmentId

In addition to the values that were already on the Meeting object, the client uses RopSetProperties to put the property values listed in the following table onto the Meeting Response object.

property	property ID	property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.Schedule.meeting.Resp.Pos
PidTagSubjectPrefix	0x003D	0x001F (PtypString)	Accepted:
PidLidSideEffects	0x8002	0x0003 (PtypInteger32)	0x00001C61 (7265)
PidLidAttendeeCriticalChange	0x81B5	0x0040 (PtypTime)	0x01C87427BF62AA00 (2008/02/21 01:19:04.352)
PidLidIsSilent	0x81E6	0x000B (PtypBoolean)	0x01 (TRUE)

The client adds the organizer by using RopModifyRecipients, and then sends the object via RopSubmitMessage. After the server returns a success code from submission, the client releases both the Meeting object and the Meeting Response objects with a RopRelease for each.

4.2.2.5 Receiving the Meeting Response

When Mr. John receives Mr. Saylor's response, the response can be recorded on the Meeting object in Mr. John's Calendar special folder.

To accomplish this task, the client issues RopOpenMessage to get a handle to the object, and RopGetPropertiesSpecific to get the PidTagMessageClass property. The server returns a handle to the Meeting Response object and the value for this property, which is "IPM.Schedule.meeting.Resp.Pos."

After seeing that this is a Meeting Response object, the client issues the RopOpenMessage for the Meeting object in the Calendar special folder. The server returns a handle for the Meeting object. The server also returns the set of RecipientRows as a result of opening the object. These RecipientRows need to be stored in an in-memory recipient cache so that they can be manipulated and then later replace those on the Meeting object.

The client uses RopGetPropertiesSpecific to get the following properties from the Meeting Request object, the values of which are returned by the server:

- PidTagSentRepresentingSearchKey
- PidTagSentRepresentingName
- PidTagSenderSearchKey
- PidTagSenderName
- PidLidAttendeeCriticalChange

If the PidTagSentRepresentingSearchKey and PidTagSentRepresentingName properties are available, these are used for searching for the RecipientRow. Otherwise, the PidTagSenderSearchKey and PidTagSenderName properties are used. The client looks through the RecipientRows, first attempting to find a PidTagSearchKey that matches the PidTagSentRepresentingSearchKey (or PidTagSenderSearchKey). If no match is found, then the client attempts to match the PidTagDisplayName property from the RecipientRow with PidTagSentRepresentingName (or PidTagSenderName).

If a RecipientRow is not found, a new one with recipient Type RECIP_CC is added to the in-memory recipient cache to represent this attendee. The following table lists the extra properties that are added to the in-memory RecipientRow that represents this attendee.

property	property ID	property type	Value
PidTagRecipientTrackStatus	0x5FFF	0x0003 (PtypInteger32)	0x00000003 (respAccepted)
PidTagRecipientTrackStatusTime	0x5FFB	0x0040 (PtypTime)	0x01C87427BCCA9A00 (2008/02/21 01:19:00.000)*

* = The value of the PidLidAttendeeCriticalChange property is rounded down to the nearest minute, then set as the value of the PidTagRecipientTrackStatusTime property.

The client uses RopRemoveAllRecipients to delete all the recipients from the Meeting object, and then uses RopModifyRecipients to copy the in-memory recipient cache back onto the Message object.

The client sets the property listed in the following table on the Meeting Request object by using RopSetProperties, followed by RopSaveChangesMessage.

property	property ID	property type	Value
PidTagProcessed	0x7D01	0x000B (PtypBoolean)	0x01 (TRUE)

Finally, the client releases both the Meeting object and Meeting Response object by using RopRelease.

4.2.2.6 Creating and Sending the Exception

Mr. John will be out of the office one Tuesday, and therefore wants to move that instance to a Wednesday. He creates an exception for this instance, adds some comments in the object body as to why it is being changed, and then sends a Meeting Update object to notify Mr. Saylor of the new date.

To accomplish this task, the client uses RopOpenMessage to open the Meeting object from Mr. John's Calendar special folder, to which the server returns a success code and a handle to the Meeting object.

The data for the exception is written to an Embedded Message object in an Attachment object on the Meeting object. A client first uses RopCreateAttachment to create the Attachment object. A server returns a success code and a handle to the new Attachment object. The property listed in the following table is set on the Attachment object.

Property type	Property	Property ID	Value
0x0003 (PtypInteger32)	PidTagAttachMethod	0x3705	0x00000005 (ATTACH_EMBEDDED_MSG)

After setting the attachment method, the client uses RopOpenEmbeddedMessage with the OpenModeFlag of Create (see [MS-OXCMSG]) to request a new Embedded Message object from the Attachment object. The server returns a success code and a handle to the new Embedded Message object. The client then uses RopSetProperties to set the properties listed in the following table on the Exception Embedded Message object.

Property	Property ID	Property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.OLE.class.{00061055-0000-0000-C000-000000000046}
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000002 (2)
PidLidAppointmentStartWhole	0x81B2	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidAppointmentEndWhole	0x81AC	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)

Property	Property ID	Property type	Value
PidLidAppointmentTimeZoneDefinitionStartDisplay	0x83A8	0x0102 (PtypBinary)	*1
PidLidAppointmentTimeZoneDefinitionEndDisplay	0x83A8	0x0102 (PtypBinary)	*1
PidLidAppointmentDuration	0x81A9	0x0003 (PtypInteger32)	0x0000001E (30)
PidLidAppointmentSubType	0x8120	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidExceptionReplaceTime	0x83AC	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)
PidLidFInvited	0x81DA	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidFExceptionalBody	0x82D8	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidClipStart	0x81BA	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidClipEnd	0x81B9	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidToAttendeesString	0x82D9	0x001F (PtypString)	desaylor
PidLidReminderTime	0x8005	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidCommonStart	0x81BC	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidCommonEnd	0x81BB	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidOwnerCriticalChange	0x8128	0x0040 (PtypTime)	0x01C874289289D700 (2008/02/21 01:24:58.608)
PidLidMeetingType	0x8314	0x0003 (PtypInteger32)	0x00010000 (65536)
PidTagStartDate	0x0060	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)

Property	Property ID	Property type	Value
PidTagEndDate	0x0061	0x0040 (PtypTime)	0x01C88EA20AF91000 (2008/03/25 18:00:00.000)
PidTagOwnerAppointmentId	0x0062	0x0003 (PtypInteger32)	0x4D9427D8 (1301555160)
best body properties	A body stream, the text of which was written by Mr. John. See [MS-OXBBODY] for details.		

*1 = The start and end dates for this appointment are both set in the same time zone. See section 4.1.4 for a description of the **TimeZoneDefinition** BLOB. The following is the value for this exception (and is the same as the associated Meeting object):


```
cb: 184
lpb:
0201300002001500500061006300690066006900630020005300740061006E00640061007200640020005400690006
D006500020002013E000000D6070000000000000000000000000000E0010000000000000C4FFFFFF00000A00000005
00020000000000000000000004000000010002000000000000000002013E000200D707000000000000000000000000000
0E0010000000000000C4FFFFFF00000B000000010002000000000000000000000300000002000200000000000000
```


The client uses RopModifyRecipients to add all the recipients from the Meeting object onto the Exception Embedded Message object, and then saves the new Exception Embedded Message object by using RopSaveChangesMessage, to which the server returns success codes.

The client uses RopSetProperties to set the properties listed in the following tableon the Exception Attachment object (not the Exception Embedded Message object).

Property	Property ID	Property type	Value
PidTagExceptionStartTime	0x7FFB	0x0040 (PtypTime)	0x01C88F2C5821C400 (2008/03/26 10:30:00.000)
PidTagExceptionEndTime	0x7FFC	0x0040 (PtypTime)	0x01C88F308903F800 (2008/03/26 11:00:00.000)
PidTagExceptionReplaceTime	0x7FF9	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)
PidTagAttachmentFlags	0x7FFD	0x0003 (PtypInteger32)	0x00000002 (afException)
PidTagAttachmentHidden	0x7FFE	0x000B (PtypBoolean)	0x01 (TRUE)

The client uses RopSaveChangesAttachment to save the changes to the Attachment object.

The client needs to use RopCreateMessage to create a new Meeting Request object in the Outbox special Folder so that attendees can be notified of the change. The server returns a success code and a handle to a new Message object.

Next, the client uses RopSetProperties to set the properties listed in the following tableon this new Meeting Request object.

Property	Property ID	Property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.Schedule.Meeting.Request
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000001 (1)
PidLidAppointmentColor	0x82CA	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidIntendedBusyStatus	0x81E2	0x0003 (PtypInteger32)	0x00000002 (2)
PidLidLocation	0x8009	0x001F (PtypString)	Your Office
PidLidRecurring	0x81FD	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidAppointmentStartWhole	0x81B2	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidAppointmentEndWhole	0x81AC	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidTimeZoneStruct	0x8214	0x0102 (PtypBinary)	*1
PidLidAppointmentTimeZoneDefinitionStartDisplay	0x83A8	0x0102 (PtypBinary)	*2
PidLidAppointmentTimeZoneDefinitionEndDisplay	0x83A9	0x0102 (PtypBinary)	*2
PidLidAppointmentTimeZoneDefinitionRecur	0x83AA	0x0102 (PtypBinary)	*3
PidLidAppointmentDuration	0x81A9	0x0003 (PtypInteger32)	0x0000001E (30)
PidLidAppointmentAuxiliaryFlags	0x82D2	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidAppointmentSubType	0x8120	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidAppointmentStateFlags	0x81B3	0x0003 (PtypInteger32)	0x00000003 (3)

Property	Property ID	Property type	Value
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000005 (respNotResponded)
PidLidAppointmentNotAllowPropose	0x82D5	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidFExceptionalAttendees	0x82D7	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidFExceptionalBody	0x82D8	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidRecurrenceType	0x81FE	0x0003 (PtypInteger32)	0x00000002 (2)
PidLidRecurrencePattern	0x81FC	0x001F (PtypString)	Every Tuesday from 10:30 A.M. to 11:00 A.M.
PidLidTimeZoneDescription	0x8213	0x001F (PtypString)	(GMT-08:00) Pacific Time (US & Canada)
PidLidClipStart	0x81BA	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidClipEnd	0x81B9	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidAllAttendeesString	0x81A8	0x001F (PtypString)	desaylor
PidLidToAttendeesString	0x82D9	0x001F (PtypString)	desaylor
PidLidAppointmentSequence	0x81AF	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidAppointmentSequenceTime	0x82E7	0x0040 (PtypTime)	0x01C874276FF4F450 (2008/02/21 01:16:51.093)
PidLidChangeHighlight	0x82EC	0x0003 (PtypInteger32)	0x00000083 (131)
PidLidReminderDelta	0x81FF	0x0003 (PtypInteger32)	0x5AE980E1 (1525252321)
PidLidReminderTime	0x8005	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidReminderSignalTime	0x8006	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidCommonStart	0x81BC	0x0040	0x01C88F6704809C00

Property	Property ID	Property type	Value
		(PtypTime)	(2008/03/26 17:30:00.000)
PidLidCommonEnd	0x81BB	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidReminderSet	0x8004	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidSideEffects	0x8002	0x0003 (PtypInteger32)	0x00001C61 (7265)
PidLidAttendeeCriticalChange	0x81B5	0x0040 (PtypTime)	0x01C8742891F14080 (2008/02/21 01:24:57.608)
PidLidWhere	0x8219	0x001F (PtypString)	Your Office
PidLidGlobalObjectId	0x81E0	0x0102 (PtypBinary)	*4
PidLidCleanGlobalObjectId	0x81B8	0x0102 (PtypBinary)	*5
PidLidAppointmentMessageClass	0x8311	0x001F (PtypString)	IPM.Appointment
PidLidIsRecurring	0x81E5	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidIsException	0x81E4	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidTimeZone	0x8212	0x0003 (PtypInteger32)	0x0000000D (13)
PidLidCalendarType	0x81B7	0x0003 (PtypInteger32)	0x00000001 (1)
PidLidOwnerCriticalChange	0x8128	0x0040 (PtypTime)	0x01C874289289D700 (2008/02/21 01:24:58.608)
PidLidMeetingType	0x8314	0x0003 (PtypInteger32)	0x00010000 (65536)
PidLidOldLocation	0x8316	0x001F (PtypString)	(null)
PidLidOldWhenStartWhole	0x83CC	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)
PidLidOldWhenEndWhole	0x83CD	0x0040 (PtypTime)	0x01C88EA20AF91000 (2008/03/25 18:00:00.000)

Property	Property ID	Property type	Value
PidTagResponseRequested	0x0063	0x000B (PtypBoolean)	0x01 (TRUE)
PidTagStartDate	0x0060	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidTagEndDate	0x0061	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidTagOwnerAppointmentId	0x0062	0x0003 (PtypInteger32)	0x4D9427D8
best body properties	A body stream, the text of which is the downlevel text, as specified in section 2.2.5.12, followed by a copy of the body text from the Exception Embedded Message object.		

*1 = See section 4.1.5 for a description of the PidLidTimeZoneStruct BLOB. The following is the value for this Meeting Request object:

```
cb: 48

lpb:
E00100000000000C4FFFFFF000000000B0000000100020000000000000000000030000000200020000000000
000
```

*2 = The PidLidAppointmentTimeZoneDefinitionRecur dates for this appointment are both set in the same time zone. See section 4.1.4 for a description of the TimeZoneDefinition BLOB. The only difference between this BLOB and that in PidLidAppointmentTimeZoneDefinitionStartDisplay / PidLidAppointmentTimeZoneDefinitionEndDisplay is that the TZRULE_FLAG_RECUR_CURRENT_TZREG flag is set in this BLOB. The following is the value for this Meeting Request object:

```
cb: 184

lpb:
02013000020015005000610063006900660069006300200053007400610 06E006400610072006400200054006900 6
D00650002000 2013E000000D607000000000000000000000000000000E00100000000000C4FFFFFF00000A00000005
000200000000000000000004000000010002000000000000002013E000300D707000000000000000000000000000000
0E00100000000000C4FFFFFF00000B000000010002000000000000000000030000000200020000000000000
```

*3 = The start and end dates for this appointment are both set in the same time zone. See section 4.1.4 for a description of the **TimeZoneDefinition** BLOB. The following is the value for this Meeting Request object:

```
cb: 184
lpb:
02013000020015005000610063006900660069006300200053007400610 06E006400610072006400200054006900 6
D00650002000 2013E000000D607000000000000000000000000000000E00100000000000C4FFFFFF00000A00000005
```

```
000200000000000000000004000000010002000000000000002013E000200D707000000000000000000000000000
0E001000000000000C4FFFFFF00000B000000010002000000000000000003000000020002000000000000
```

*4 = The following is the value of the PidLidGlobalObjectId for this Meeting Request object. See section 4.1.2 for a description of the Global Obj ID BLOB.

```
cb: 56
lpb:
040000008200E00074C5B7101A82E00807D803195025D461E473C80100000000000000000100000002A5844B3A444F
74A9C246C60886F116B
```

*5 = The following is the value of the PidLidCleanGlobalObjectId for this Meeting Request object. This is identical to the value of the PidLidGlobalObjectId property, except that the *Year*, *Month*, and *Day* fields are filled with zeros.

```
cb: 56
lpb:
040000008200E00074C5B7101A82E00800000000005025D461E473C80100000000000000000100000002A5844B3A444F
74A9C246C60886F116B
```

In addition to these properties, the client needs to use RopSetProperties to add all properties that are required to send a Message object, as specified in [MS-OXOMSG], to the Meeting Request object so that it can be delivered to the attendee. This client also needs to use RopModifyRecipients to add a RecipientRow for Mr. Saylor to the Meeting Request object.

Now that the Meeting Request object has been created and filled in, it will be sent instead of saved. The client uses RopSubmitMessage to send this Message object for transport.

The client makes the changes listed in the following table to the Meeting object (the object that represents the recurring series) on Mr. John's calendar by using RopSetProperties.

Property	Property ID	Property type	Value
PidLidAppointmentRecur	0x81AD	0x0102 (PtypBinary)	*1
PidLidFExceptionalAttendees	0x82D7	0x000B (PtypBoolean)	0x01 (TRUE)

*1 = The value of the PidLidAppointmentRecur property will include necessary information about this new exception. The following is the new value for this Meeting object:

```
cb: 114

lpb:
043004300B2001000000C021000001000000000000000400000023200000A000000000000000100000A025C40C0
1000000402BC40C2088C30CDF80E95A0630000009300000760200009402000001 00B62DC40CD42DC40C1628C40C00
020000000004000000000000000000000000000000
```

Finally, the client issues RopSaveChangesMessage to save the Meeting object that represents the recurring series, and then uses RopRelease to release all handles (**embedded message**, attachment, meeting, and Meeting Request objects).

4.2.2.7 Accepting the Exception

After receiving the Meeting Update object, Mr. Dennis Saylor decides that the change will still work with his schedule. The Calendar object in Mr. Saylor's Calendar folder needs to be updated, and a Meeting Response object needs to be sent back to Mr. John.

To accomplish this task, the client uses RopOpenMessage to open the Meeting Update object to which the server returns a success code and a handle. The client uses RopGetPropertiesSpecific to get at least the following properties: PidTagOwnerAppointmentId, PidLidGlobalObjectId, and PidLidCleanGlobalObjectId.

The client uses RopGetContentsTable to open the **contents table** of the Calendar special folder. The server returns a handle to the contents table. The client sets at least the following **column set** on the contents table by using RopSetColumns:

- PidTagMid
- PidTagOwnerAppointmentId
- PidLidGlobalObjectId

The Meeting Update object in this example has a value for the PidTagOwnerAppointmentId property, so the client uses RopSortTable to sort the contents table in ascending order of this property. The client then uses RopFindRow to find the first matching table row. The server returns a success code with the first matching row, or returns an error code if a matching row was not found.

For each matching row, the client compares the value of the PidLidCleanGlobalObjectId property from the Meeting Update object with the value of the PidLidGlobalObjectId property in the row, until a match is found.<132> After finding a matching row, the client issues RopOpenMessage by using the value of the PidTagMid property from that row to open the Meeting object, to which the server returns a success code and a handle.

Having obtained the recurring series, the client tries to find the Exception Attachment object. The client uses RopGetAttachmentTable to open the list of attachments. The client uses RopSetColumns to set at least the following columns on this table:

- PidTagAttachMethod
- PidTagAttachmentFlags
- PidTagAttachNumber
- PidTagExceptionReplaceTime

The client uses RopQueryRows to loop through the rows in the **attachment table**, attempting to find the matching Exception Attachment object. If the value of the PidTagAttachmentFlags property in a row does not include the *afException* flag, the attachment does not represent an exception. To find the matching Exception Attachment object, the client uses the values of the *Day*, *Month*, and *Year* fields of the PidLidGlobalObjectId property on the Meeting Update object to compute the replace date/time, and looks for an Exception Attachment object with a matching value.<133>

In this example, an Exception Attachment object does not exist, so the client uses RopCreateAttachment to create a new one, to which the server returns a success code and a handle. The client uses RopSetProperties to set the following property on the Attachment object.

Property	Property ID	Property type	Value
PidTagAttachMethod	0x3705	0x0003 (PtypInteger32)	0x00000005 (ATTACH_EMBEDDED_MSG)

After setting the attachment method, the client uses RopOpenEmbeddedMessage with the OpenModeFlag of Create (see [MS-OXCMSG]) to request a new Embedded Message object from the Attachment object. The server returns a success code and a handle to the new Embedded Message object. The client then uses RopSetProperties to set the properties listed in the following table on the Exception Embedded Message object, as copied from the Meeting Request object :

Property	Property ID	Property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.OLE.class.{00061055-0000-0000-C000-000000000046}
PidTagSubjectPrefix	0x003D	0x001F (PtypString)	
PidTagNormalizedSubject	0x0E1D	0x001F (PtypString)	Weekly meeting
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000001 (olTentative)
PidLidIntendedBusyStatus	0x81E2	0x0003 (PtypInteger32)	0x00000002 (olBusy)
PidLidLocation	0x8009	0x001F (PtypString)	Your Office
PidLidRecurring	0x81FD	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidAppointmentStartWhole	0x81B2	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidAppointmentEndWhole	0x81AC	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidTimeZoneStruct	0x8214	0x0102 (PtypBinary)	*1
PidLidAppointmentTimeZoneDefinitionStartDisplay	0x83A8	0x0102 (PtypBinary)	*2
PidLidAppointmentTimeZoneDefinitionEndDisplay	0x83A9	0x0102 (PtypBinary)	*2
PidLidAppointmentTimeZoneDefinitionRecur	0x83AA	0x0102 (PtypBinary)	*3

Property	Property ID	Property type	Value
PidLidAppointmentDuration	0x81A9	0x0003 (PtypInteger32)	0x0000001E (30)
PidLidAppointmentAuxiliaryFlags	0x82D2	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidAppointmentSubType	0x8120	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidAppointmentStateFlags	0x81B3	0x0003 (PtypInteger32)	0x00000003 (3)
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000005 (respNotResponded)
PidLidAppointmentNotAllowPropose	0x82D5	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidExceptionReplaceTime	0x83AC	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)
PidLidFInvited	0x81DA	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidFExceptionalAttendees	0x82D7	0x000B (PtypBoolean)	0x00 (FALSE)
PidLidFExceptionalBody	0x82D8	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidRecurrenceType	0x81FE	0x0003 (PtypInteger32)	0x00000002 (2)
PidLidRecurrencePattern	0x81FC	0x001F (PtypString)	Every Tuesday from 10:30 A.M. to 11:00 A.M.
PidLidTimeZoneDescription	0x8213	0x001F (PtypString)	(GMT-08:00) Pacific Time (US & Canada)
PidLidClipStart	0x81BA	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidClipEnd	0x81B9	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidAllAttendeesString	0x81A8	0x001F (PtypString)	desaylor
PidLidToAttendeesString	0x82D9	0x001F	desaylor

Property	Property ID	Property type	Value
		(PtypString)	
PidLidAppointmentSequence	0x81AF	0x0003 (PtypInteger32)	0x00000000 (0)
PidLidAppointmentSequenceTime	0x82E7	0x0040 (PtypTime)	0x01C874276FF4F450 (2008/02/21 01:16:51.093)
PidLidChangeHighlight	0x82EC	0x0003 (PtypInteger32)	0x00000083 (131)
PidLidReminderTime	0x8005	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidCommonStart	0x81BC	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidLidCommonEnd	0x81BB	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidLidAttendeeCriticalChange	0x81B5	0x0040 (PtypTime)	0x01C8742891F14080 (2008/02/21 01:24:57.608)
PidLidWhere	0x8219	0x001F (PtypString)	Your Office
PidLidGlobalObjectId	0x81E0	0x0102 (PtypBinary)	*4
PidLidCleanGlobalObjectId	0x81B8	0x0102 (PtypBinary)	*5
PidLidAppointmentMessageClass	0x8311	0x001F (PtypString)	IPM.appointment
PidLidIsRecurring	0x81E5	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidIsException	0x81E4	0x000B (PtypBoolean)	0x01 (TRUE)
PidLidTimeZone	0x8212	0x0003 (PtypInteger32)	0x0000000D (13)
PidLidCalendarType	0x81B7	0x0003 (PtypInteger32)	0x00000001 (CAL_GREGORIAN)
PidLidOwnerCriticalChange	0x8128	0x0040	0x01C874289289D700

Property	Property ID	Property type	Value
		(PtypTime)	(2008/02/21 01:24:58.608)
PidLidMeetingType	0x8314	0x0003 (PtypInteger32)	0x00010000 (65536)
PidLidOldLocation	0x8316	0x001F (PtypString)	(null)
PidLidOldWhenStartWhole	0x83CC	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)
PidLidOldWhenEndWhole	0x83CD	0x0040 (PtypTime)	0x01C88EA20AF91000 (2008/03/25 18:00:00.000)
PidTagResponseRequested	0x0063	0x000B (PtypBoolean)	0x01 (TRUE)
PidTagStartDate	0x0060	0x0040 (PtypTime)	0x01C88F6704809C00 (2008/03/26 17:30:00.000)
PidTagEndDate	0x0061	0x0040 (PtypTime)	0x01C88F6B3562D000 (2008/03/26 18:00:00.000)
PidTagOwnerAppointmentId	0x0062	0x0003 (PtypInteger32)	0x4D9427D8
best body properties	The client can look for and remove the downlevel text, as specified in section 2.2.5.12, before copying the text stream onto the new Exception Embedded Message object.		

*1 = See section 4.1.5 for a description of the PidLidTimeZoneStruct BLOB. The following is the value for this Meeting Request object:

```
cb: 48

lpb:
E00100000000000C4FFFFFF000000000B0000000100020000000000000000000030000000200020000000000
000
```

*2 = The PidLidAppointmentTimeZoneDefinitionRecur dates for this appointment are both set in the same time zone. See section 4.1.4 for a description of the TimeZoneDefinition BLOB. The only difference between this BLOB and that in PidLidAppointmentTimeZoneDefinitionStartDisplay / PidLidAppointmentTimeZoneDefinitionEndDisplay is that the TZRULE_FLAG_RECUR_CURRENT_TZREG flag is set in this BLOB. The following is the value for this Meeting Request object:

```
cb: 184

lpb:
02013000020015005000610063006900660069006300200053007400610060064006100720064002000540069006
D006500020002013E000000D607000000000000000000000000000000E001000000000000C4FFFFFF00000A00000005
000200000000000000000000400000001000200000000000000002013E000300D707000000000000000000000000
0E001000000000000C4FFFFFF00000B000000010002000000000000000000030000000200020000000000000
```

*3 = The start and end dates for this appointment are both set in the same time zone. See section 4.1.4 for a description of the **TimeZoneDefinition** BLOB. The following is the value for this Meeting Request object:

```
cb: 184
lpb:
02013000020015005000610063006900660069006300200053007400610060064006100720064002000540069006
D006500020002013E000000D607000000000000000000000000000000E001000000000000C4FFFFFF00000A00000005
000200000000000000000000400000001000200000000000000002013E000200D707000000000000000000000000
0E001000000000000C4FFFFFF00000B000000010002000000000000000000030000000200020000000000000
```

*4 = The following is the value of the PidLidGlobalObjectId property for this Meeting Request object. See section 4.1.2 for a description of the Global Obj ID BLOB.

```
cb: 56
lpb:
040000008200E00074C5B7101A82E00807D803195025D461E473C801000000000000000001000000002A5844B3A444F
74A9C246C60886F116B
```

*5 = The following is the value of the PidLidCleanGlobalObjectId property for this Meeting Request object. This is identical to the value of the PidLidGlobalObjectId property except that the *Year*, *Month*, and *Day* fields are filled with zeros.

```
cb: 56
lpb:
040000008200E00074C5B7101A82E00800000000005025D461E473C801000000000000000001000000002A5844B3A444F
74A9C246C60886F116B
```

The client uses RopModifyRecipients to set the recipients on the Exception Embedded Message object. The recipients are obtained from the **RecipientRows** of the Meeting Request object, as well as the PidLidAppointmentUnsendableRecipients property. In addition, if the organizer (in this case, Mr. John) is not in the list of recipients, his information is obtained from the PidTagSentRepresentingSearchKey and PidTagSentRepresentingName properties and added as a **RecipientRow**. The Exception Embedded Message object is saved by using RopSaveChangesMessage, to which the server returns a success code.

After saving the Exception Embedded Message object, the client uses RopSetProperties to set the properties listed in the following table on the Exception Attachment object (not the Exception Embedded Message object).

Property	Property ID	Property type	Value
PidTagExceptionStartTime	0x7FFB	0x0040 (PtypTime)	0x01C88F2C5821C400 (2008/03/26 10:30:00.000)
PidTagExceptionEndTime	0x7FFC	0x0040 (PtypTime)	0x01C88F308903F800 (2008/03/26 11:00:00.000)
PidTagExceptionReplaceTime	0x7FF9	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)
PidTagAttachmentFlags	0x7FFD	0x0003 (PtypInteger32)	0x00000002 (afException)
PidTagAttachmentHidden	0x7FFE	0x000B (PtypBoolean)	0x01 (TRUE)

The client uses RopSaveChangesAttachment to save the changes to the Attachment object.

Now that the exception has been created, the client makes the following change to the Meeting object (the object that represents the recurring series) on Mr. Saylor's calendar by using RopSetProperties.

Property	Property ID	Property type	Value
PidLidAppointmentRecur	0x81AD	0x0102 (PtypBinary)	*1

*1 = The value of the PidLidAppointmentRecur property will include necessary information about this new exception. The following is the new value for the attendee's Meeting object.

```
cb: 114

lpb:
043004300B200100000C02100001000000000000000400000023200000A000000000000000010000000A025C40C0
1000000402BC40C2088C30CDF80E95A0630000009300000760200009402000001 00B62DC40CD42DC40C1628C40C00
020000000004000000000000000000000000000000
```

The client sets the following property on the Meeting Request object by using RopSetProperties, followed by RopSaveChangesMessage.

Property	Property ID	Property type	Value
PidTagProcessed	0x7D01	0x000B (PtypBoolean)	0x01 (TRUE)

After processing the Meeting Request object, the client is now ready to act on the response. To start, the changes listed in the following table are made to the Exception Embedded Message object by using RopSetProperties.

property	Property ID	Property type	Value
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000002 (2)

property	Property ID	Property type	Value
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000003 (respAccepted)
PidLidAppointmentReplyTime	0x8139	0x0040 (PtypTime)	0x01C87428FEA81000 (2008/02/21 01:28:00.000)
PidLidAppointmentReplyName	0x81AE	0x001F (PtypString)	desaylor

The client again saves the Exception Embedded Message object by using RopSaveChangesMessage and another RopSaveChangesMessage to save the Meeting object that represents the recurring series, to which the server returns success codes.

The last thing the client needs to do is send a response to the organizer. The client creates a new Meeting Response object in the Outbox special Folder by using RopCreateMessage, to which the server returns a success code and a handle. The client sets the following properties on this new Message object by using RopSetProperties using the values from the Exception Embedded Message object:

- PidTagNormalizedSubject
- PidLidBusyStatus
- PidLidAppointmentColor
- PidLidLocation
- PidLidRecurring
- PidLidAppointmentStartWhole
- PidLidAppointmentEndWhole
- PidLidAppointmentTimeZoneDefinitionStartDisplay
- PidLidAppointmentTimeZoneDefinitionEndDisplay
- PidLidAppointmentDuration
- PidLidAppointmentAuxiliaryFlags
- PidLidAppointmentSubType
- PidLidAppointmentRecur
- PidLidRecurrenceType
- PidLidRecurrencePattern
- PidLidTimeZoneStruct
- PidLidAppointmentTimeZoneDefinitionRecur
- PidLidTimeZoneDescription
- PidLidClipStart

- PidLidClipEnd
- PidLidAppointmentSequence
- PidLidCommonStart
- PidLidCommonEnd
- PidLidWhere
- PidLidGlobalObjectId
- PidLidCleanGlobalObjectId
- PidLidAppointmentMessageClass
- PidLidIsRecurring
- PidLidIsException
- PidLidTimeZone
- PidLidCalendarType
- PidLidOwnerCriticalChange
- PidTagStartDate
- PidTagEndDate
- PidTagOwnerAppointmentId

In addition to these, the client uses RopSetProperties to put the property values listed in the following table onto the Meeting Response object.

Property	Property ID	Property type	Value
PidTagMessageClass	0x001A	0x001F (PtypString)	IPM.Schedule.meeting.Resp.Pos
PidTagSubjectPrefix	0x003D	0x001F (PtypString)	Accepted:
PidLidSideEffects	0x8002	0x0003 (PtypInteger32)	0x00001C61 (7265)
PidLidAttendeeCriticalChange	0x81B5	0x0040 (PtypTime)	0x01C874292153F290 (2008/02/21 01:28:58.169)
PidLidIsSilent	0x81E6	0x000B (PtypBoolean)	0x01 (TRUE)

The client adds the organizer by using RopModifyRecipients, and then sends the object via RopSubmitMessage. After the server returns a success code from submission, the client releases all objects, including the embedded message, attachment, attachment table, meeting, and Meeting Request objects, by using a RopRelease for each.

5 Security

5.1 Security Considerations for Implementers

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

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following product versions. References to product versions include released service packs.

- 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. If a service pack number appears with the product version, behavior changed in that service pack. The new behavior also applies to subsequent service packs of the product unless otherwise specified.

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

<1> Section 2.2: Office sets the following additional properties on a new object, regardless of user input:PidLidAgingDontAgeMe, PidLidCurrentVersion, PidLidCurrentVersionName, PidLidValidFlagStringProof, PidTagAlternateRecipientAllowed, PidTagClientSubmitTime, PidTagDeleteAfterSubmit,, PidTagMessageDeliveryTime, PidTagOriginatorDeliveryReportRequested, PidTagReadReceiptRequested

<2> Section 2.2: The following additional properties can be set on items described by the appointment and Meeting object protocol for backward compatibility with earlier clients. These properties are not used by Office:PidLidRequiredAttendees, PidLidOptionalAttendees, PidLidResourceAttendees, PidLidDelegateMail, PidLidSingleInvite, PidLidTimeZone, PidLidStartRecurrenceDate, PidLidStartRecurrenceTime, PidLidEndRecurrenceDate, PidLidEndRecurrenceTime, PidLidDayInterval, PidLidWeekInterval, PidLidMonthInterval, PidLidYearInterval, PidLidMonthOfYearMask, PidLidRecurrenceType, PidLidAllAttendeesList.

<3> Section 2.2: Outlook 2007 sets the following properties regardless of user input; their values have no meaning in the context of this protocol:PidLidTaskStatus, PidLidPercentComplete, PidLidTaskActualEffort, PidLidTaskEstimatedEffort, PidLidTaskVersion, PidLidTaskState, PidLidTaskComplete, PidLidTaskAssigner, PidLidTaskOrdinal, PidLidTaskNoCompute, PidLidTaskFRecurring, PidLidTaskRole, PidLidTaskOwnership, PidLidTaskAcceptanceState, PidLidTaskFFixOffline.

<4> Section 2.2.1.3: Exchange does not set the auxApptFlagCopied flag when copying Calendar objects.

<5> Section 2.2.1.7: Exchange 2003 ignores this property and always computes this from the difference between PidLidAppointmentEndWhole and PidLidAppointmentStartWhole.

<6> Section 2.2.1.13: Exchange 2003 does not read or write this property.

<7> Section 2.2.1.25: Outlook 2003 and Exchange 2003 instead use the following properties properties to track unsendable attendees:

PidLidNonSendableTo

PidLidNonSendableCc

PidLidNonSendableBcc

PidLidNonSendToTrackStatus

PidLidNonSendCcTrackStatus

PidLidNonSendBccTrackStatus

<8> Section 2.2.1.29: When a Meeting object is created, Office sets this value to the number of minutes between the start time and midnight, January 1, 1601. When trying to find a Meeting object, Office sorts the table according to the PidTagOwnerAppointmentId property, thus allowing increased performance in the search.

<9> Section 2.2.1.36: Office and Exchange allow the user to choose whether they want to send a Meeting Response object to the organizer.

<10> Section 2.2.1.41: PidLidAppointmentTimeZoneDefinitionRecur contains one TZRule that is marked with the TZRULE_FLAG_EFFECTIVE_TZREG flag. This TZRule has fields *lBias*, *lStandardBias*, *lDaylightBias*, *stStandardDate*, and *stDaylightDate*. If any of these fields do not match exactly the corresponding field in PidLidTimeZoneStruct, the propertiesPidLidAppointmentTimeZoneDefinitionRecur and PidLidTimeZoneStruct are considered inconsistent. If this property is not set, PidLidTimeZoneStruct will be used instead

<11> Section 2.2.1.41: Outlook 2003 does not support PidLidAppointmentTimeZoneDefinitionRecur.. The fields in this BLOB are encoded in little-endian byte order.

<12> Section 2.2.1.41: In the Windows operating system, the unique names of all currently defined time zones can be obtained by enumerating key names of all registry keys that appear as children of the following registry key: HKLM\Software\Microsoft\Windows NT\CurrentVersion\Time Zones. For example, on Windows Vista as of January 1, 2008, this list consists of the following:

Afghanistan Standard Time

Alaskan Standard Time

Arab Standard Time

Arabian Standard Time

Arabic Standard Time

Atlantic Standard Time

AUS Central Standard Time

AUS Eastern Standard Time

Azerbaijan Standard Time

Azores Standard Time

Canada Central Standard Time

Cape Verde Standard Time

Caucasus Standard Time

Cen. Australia Standard Time

Central America Standard Time

Central Asia Standard Time

Central Brazilian Standard Time

Central Europe Standard Time

Central European Standard Time

Central Pacific Standard Time

Central Standard Time

Central Standard Time (Mexico)

China Standard Time

Dateline Standard Time

E. Africa Standard Time

E. Australia Standard Time

E. Europe Standard Time

E. South America Standard Time

Eastern Standard Time

Egypt Standard Time

Ekaterinburg Standard Time

Fiji Standard Time

FLE Standard Time

Georgian Standard Time

GMT Standard Time

Greenland Standard Time

Greenwich Standard Time

GTB Standard Time

Hawaiian Standard Time

India Standard Time

Iran Standard Time

Israel Standard Time

Jordan Standard Time

Korea Standard Time

Mid-Atlantic Standard Time

Middle East Standard Time

Mountain Standard Time

Mountain Standard Time (Mexico)

Myanmar Standard Time

N. Central Asia Standard Time

Namibia Standard Time

Nepal Standard Time

New Zealand Standard Time

Newfoundland Standard Time

North Asia East Standard Time

North Asia Standard Time

Pacific SA Standard Time

Pacific Standard Time

Pacific Standard Time (Mexico)

Romance Standard Time

Russian Standard Time

SA Eastern Standard Time

SA Pacific Standard Time

SA Western Standard Time

Samoa Standard Time

SE Asia Standard Time

Singapore Standard Time

South Africa Standard Time

Sri Lanka Standard Time

Taipei Standard Time

Tasmania Standard Time

Tokyo Standard Time

Tonga Standard Time

US Eastern Standard Time

US Mountain Standard Time

Vladivostok Standard Time

W. Australia Standard Time

W. Central Africa Standard Time

W. Europe Standard Time

West Asia Standard Time

West Pacific Standard Time

Yakutsk Standard Time

<13> Section 2.2.1.44: Exchange 2003 and Exchange 2007 use the signal time rather than the start time when calculating whether exceptions overlap. Outlook 2003 and Outlook 2007 use the start time.

<14> Section 2.2.1.44.1: These values can be read by Office but are not used:NameValueDescriptionMonthEnd0x0004The event has a month end recurrence. HjMonth0x000AThe event has a monthly recurrence in the Hijri calendar. For this PatternType, the CalendarType MUST be set to 0x0000.

<15> Section 2.2.1.44.1: Exchange 2003 supports only the Gregorian calendar. Exchange 2007 does not support the CAL_SAKA calendar.

<16> Section 2.2.1.44.1: The following is a description of how the FirstDateTime value is used for a daily recurrence pattern:Daily recurrences are evaluated by advancing by the number of minutes required to reach the next instance (period). This will vary depending on the frequency/interval (every x days), but given that granularity is days, the number of minutes will always be a multiple of 1440 (number of minutes in a day).Taking a valid instance and adding the period will yield the next instance. Therefore, finding a valid instance is essential. FirstDateTime is used to find a valid day within the pattern, by computing the offset of the start clip date given the period (clipStart modulo period). This produces the number of minutes that need to be subtracted from an input date prior to checking whether it is a valid instance (it is valid if the adjusted date modulo period yields 0 (zero)). If it is not a valid instance, the modulo operation will yield the value to subtract from the input date to find a valid instance. For example, given the following dates (in minutes, assuming time is truncated so the value indicates the day), and a pattern that starts on Day 1:

Day 0 = 0

Day 1 = 1440

Day 2 = 2880

Day 3 = 4320

...

It can be seen that an "Every 1 day" (period is 1440 * 1 = 1440) pattern is uninteresting, *FirstDateTime* will always be 0 (zero), as (*Day* X modulo 1440) will always yield 0 (zero), which indicates that every input date is a valid instance in the pattern.Now consider an "Every 3 days" (period is 1440 * 3 = 4320) pattern. In this case, valid instances are 1, 4, 7, 10,..., so not every day is a part of the pattern. In this case, *FirstDateTime* will be computed to be 1440, which indicates that this offset is subtracted from an input date prior to determining if it is a valid instance. If Day 9 (12960) is the input date, the following computation determines if this is a valid instance: Adjusted input date: 12960 - 1440 = 11520Check for valid date: 11520 modulo 4320 = 2880 (this is not a valid instance, and 2880 minutes, or 2 days, needs to be subtracted to find the previous valid instance). Previous valid instance: 12960 - 2880 = 10080 (this is Day 7, and is a valid instance). An interesting aspect of *FirstDateTime* for a daily recurrence pattern is that it will always be a value between 0 (zero) and (period - 1440).

<17> Section 2.2.1.44.1: The following is a description of how the FirstDateTime value is used for a weekly recurrence pattern. Weekly recurrences are slightly more complex, as a valid week needs to be found, as well as a valid day within that week. This will vary depending on the frequency/interval (every x weeks), but will also vary by the first day of week with which the pattern was created. The first day of week dependency is what makes this somewhat more complex. For example, consider the pattern "Every 2 weeks on Monday, Tuesday, and Friday, starting in week 2." If the first day of the week is Wednesday, then when evaluating the pattern, the Monday, Tuesday, and Friday instances in a given week are not the same as they would be if the first day of week was Sunday. The following list might make this a little bit easier to understand:

Assuming a pattern "Every 2 weeks on Mon, Tue, and Fri., Starting in week 2

```
WeekFirst Day of Week Is Sunday
SuMoTuWeThFrSa
11234567
28(9)(10)1112(13)14
315161718192021
422(23)(24)2526(27)28


WeekFirst Day of Week Is Wednesday
WeThFrSaSuMoTu
145678910
21112(13)1415(16)(17)
318192021222324
42526(27)2829(30)(31)
```

If the first day of the week was Sunday, the valid dates would be the 9th, 10th, 13th, 23rd, 24th, and 27th of the month, but if the first day of the week was defined to be Wednesday, the valid dates would be the 13th, 16th, 17th, 27th, 30th, and 31st of the month. The first day of week makes a huge difference. When evaluating the weekly recurrence pattern, all instances need to be on the same week (relative to the first day of week setting).

With a better understanding of the evaluation, focus can shift to what information is trying to be preserved to properly find a valid instance given some input date. First, a valid week must be found, which is where *FirstDateTime* comes into play. After it is adjusted to a valid week, a valid day within the week can be found.

As was the case for daily, *FirstDateTime* represents the necessary offset to adjust from the input week to find a valid week. The only difference is that this offset is adjusted relative to the beginning of a week, which requires also looking at the first day of week.

To compute the offset:

1. Adjust the start clip date to the beginning of a week.

2. Compute the clip start offset (*FirstDateTime*) by taking the adjusted start clip date value modulo (period * 10080). Unlike daily patterns, Period is not stored in number of minutes, rather number of weeks. 10080 is the number of minutes in a week (1440 * 7). Because this value is adjusted to beginning of the week, and because 1-based computations will be used, the value of *FirstDateTime* will always be 1440 (1 day) less than what one might expect.

For example:

8640 instead of 10080 for 1 week.

18720 instead of 20160 for 2 weeks.

After finding a valid week, the first valid day in the week is found.

Using the previous example (week starts on Wednesday), assume that the input date provided was the 21st.

1. Adjust to the start of the week, which is the 18th.

2. Using the FirstDateTime weekly offset value, determine if this is a valid week. If it is not, this computation will provide the number of weeks to advance to get to a valid week. In the example, this would adjust the week to the 25th.

3. Look forward until a valid day is found, which would be the 27th, the next valid instance.

<18> Section 2.2.1.44.1: The following is a description of how the *FirstDateTime* value is used for a Monthly or Yearly recurrence pattern.

Monthly and Yearly are evaluated in the same way; yearly just happens to be a monthly pattern that occurs every 12 months.

With an understanding of how the *FirstDateTime* value is used in a daily pattern, the monthly/yearly pattern is straightforward. *FirstDateTime* is the offset (in months relative to 1600) needed to find a valid month within the recurrence.

From an input date, the next valid month is found by adding the difference between the input month and the 1600 offset (*FirstDateTime*) modulo period.

Other details exist for non-Gregorian calendars, which can have leap months and other non-Gregorian specific details.

<19> Section 2.2.1.44.1: Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 always write a default value of 0x0000000A for the Occurrence count when the recurrence pattern has no end date.

<20> Section 2.2.1.44.1: Exchange 2007 does not allow duplicate entries, and will remove them if they are present.

<21> Section 2.2.1.44.1: Exchange 2007 does not allow duplicate entries, and will remove them if they are present.

<22> Section 2.2.1.44.2: This flag is not set in Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2007. This flag is reserved for future enhancements and is not used.

<23> Section 2.2.1.44.3: This field does not exist in Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2007. This field is reserved for future enhancements and is not used.

<24> Section 2.2.1.47: Outlook 2007 sets this property, but Outlook 2003, Exchange 2003, and Exchange 2007 do not.

<25> Section 2.2.1.49: Exchange 2003 does not read or write this property, but Outlook 2003, Outlook 2007, and Exchange 2007 do.

<26> Section 2.2.1.50: Outlook 2003 reads and writes the properties in this section. Outlook 2007 does not write any of these properties but reads some of them. Exchange 2003 and Exchange 2007 do not read or write these properties.

<27> Section 2.2.2: Calendar objects can also have the following reminder-related properties as specified in [MS-OXORMDR]:PidLidReminderSet, PidLidReminderSignalTime, PidLidReminderDelta, PidLidReminderTime, PidLidReminderOverride, PidLidReminderPlaySound, PidLidReminderFileParameter.

<28> Section 2.2.2.2: Exchange 2003 only includes the seCoerceToInbox and seOpenForCtxMenu flags. Without all the flags, the Outlook user interface will not always behave as expected when a Calendar object is moved, deleted, or copied, or when a context menu is displayed for the object.

<29> Section 2.2.2.3: The PidLidFExceptionalAttendees property is used to determine, from an Appointment object, whether attendees have been invited to any exceptions.

<30> Section 2.2.3: Meeting objects can also have the following property: PidLidOriginalStoreEntryId.

<31> Section 2.2.3.8: If there is more than one resource in a Meeting object, the PidLidLocation property is set to the first sendable resource that is added to the meeting. If none of the resources are sendable, the PidLidLocation property is set to the first unsendable resource that is added to the meeting

<32> Section 2.2.3.9.1: Outlook 2003 and Outlook 2007 use these reserved flags for internal information that does not affect the appointment and Meeting object protocol. A server or non-Office clients do not need to read these flags, but need to keep the values if they are set.

<33> Section 2.2.3.9.1: Outlook 2003 and Outlook 2007 use these reserved flags for internal information that does not affect the appointment and Meeting object protocol. A server or non-Office clients do not need to read these flags, but need to keep the values if they are set.

<34> Section 2.2.3.9.1: Outlook 2003 and Outlook 2007 use these reserved flags for internal information that does not affect the appointment and Meeting object protocol. A server or non-Office clients do not need to read these flags, but need to keep the values if they are set.

<35> Section 2.2.4.2: If this value is not specified, Exchange 2003 will assume the last modified time as this value. Exchange 2007, Outlook 2003, and Outlook 2007 do not make this assumption.

<36> Section 2.2.4.3: Exchange 2003 does not read or write this property.

<37> Section 2.2.4.6: The data in this table is used by Outlook 2003 and Outlook 2007, although its content is subject to change with future time zone updates.

<38> Section 2.2.5: Meeting Request objects and Meeting Update objects can also have the following properties, which have no effect on the Appointment and Meeting object protocol: PidLidTrustRecipientHighlights.

<39> Section 2.2.5.2: Exchange 2003 and Outlook 2003 do not read or write this property.

<40> Section 2.2.5.3: The property PidLidForwardInstance is used by Outlook 2003, but not by Outlook 2007, Exchange 2003, or Exchange 2007.

<41> Section 2.2.5.7: Outlook 2007 and Exchange 2007 set this property, but Outlook 2003 and Exchange 2003 do not.

<42> Section 2.2.5.8: Outlook 2007 and Exchange 2007 set this property, but Outlook 2003 and Exchange 2003 do not.

<43> Section 2.2.5.9: Outlook 2003 and Exchange 2003 set this property, but Outlook 2003 and Exchange 2003 do not.

<44> Section 2.2.5.12: Outlook 2003 and Outlook 2007 show the values of the PidLidAppointmentStartWhole, PidLidAppointmentEndWhole, and PidLidLocation properties as the downlevel text. Exchange 2003 and Exchange 2007 do not add downlevel text.

<45> Section 2.2.6.2: .For English, Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 use the string "New Time Proposed" to indicate that the Meeting Response object includes a new date/time proposal. If no proposal is included, Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 use "Accepted," "Tentative," or "Declined" for an accepted, tentatively accepted, or declined meeting response, respectively.

<46> Section 2.2.7.2: For English, Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 use the string "Canceled".

<47> Section 2.2.9: There are some circumstances in which the number of Exception Attachment objects will not match the number of values in the ModifiedInstanceDates field of the PidLidAppointmentRecur property. It can happen in the following case: When an Exception Attachment object cannot be found in the set of attachments, a client or server can create it. In some cases, this erroneously leads to multiple Exception Attachment objects for one instance.

<48> Section 2.2.9.1.4: If the user changes the client computer's time zone after this property is written, the value of this property will no longer match what is expected by the client. Therefore, a client or sever cannot rely on this property to be correct.

<49> Section 2.2.9.1.5: If the user changes the client computer's time zone after this property is written, the value of this property will no longer match what is expected by the client. Therefore, a client or sever cannot rely on this property to be correct.

<50> Section 2.2.9.1.6: Outlook 2003 and Outlook 2007 do not write this value.

<51> Section 2.2.10: An end user can create calendar items in any Calendar folder. However, free/busy information is only calculated from the Calendar special folder.

<52> Section 2.2.11.1: Exchange 2010 supports public folder referrals, but does not support public folders when client connection services are deployed on an Exchange server that does not also have a Mailbox store installed.

<53> Section 3.1.4.1: When an end user creates a meeting in a Calendar folder other than the Calendar special folder, Outlook 2003 and Outlook 2007 will ask the user if he or she wants to create a clone in the Calendar special folder. Exchange 2003 and Exchange 2007 will not create a clone of the meeting.

<54> Section 3.1.4.3: A copy of a Calendar object is a static copy of the original. When the source object is a meeting, the new copy will not be updated with any future changes made by the organizer.

<55> Section 3.1.4.3: Outlook 2007 sets this property, but Outlook 2003 and Exchange 2003 and Exchange 2007 do not.

<56> Section 3.1.4.3: Outlook 2003 and Outlook 2007 sometimes do not copy the recipient list. If the RecipientRows from a Meeting object are not copied, the resulting snapshot will not show who was invited to the meeting at the time the copy was made.

<57> Section 3.1.4.3: Outlook 2007 sets this property, but Outlook 2003, Exchange 2003 and Exchange 2007 do not.

<58> Section 3.1.4.3: Outlook 2007 sets this property, but Outlook 2003, Exchange 2003 and Exchange 2007 do not.

<59> Section 3.1.4.4: Outlook 2007 and Exchange 2007 require the organizer to send a meeting cancellation to attendees when deleting a meeting. Outlook 2003 and Exchange 2003 give the user an option to delete without sending a cancellation.

<60> Section 3.1.4.6.1.1: Outlook 2003 and Outlook 2007 attempt direct booking only for resources. Exchange 2003 and Exchange 2007 do not attempt direct booking for any attendees.

<61> Section 3.1.4.6.1.1: This requires public folders to be enabled on the server. Exchange 2007 allows a configuration without public folders, in which case direct booking would not be possible.

<62> Section 3.1.4.6.2: Outlook 2007 and Exchange 2007support the calendar dictionary, but Outlook 2003and Exchange 2003do not.

<63> Section 3.1.4.6.2.1: Outlook 2007 skips automatic creation of the Meeting object based on the values of these properties, but Outlook 2003, Exchange 2007, and Exchange 2003 SP2 do not.

<64> Section 3.1.4.6.2.2: Outlook 2003 and Outlook 2007 do this in certain circumstances. Exchange 2003 SP2 and Exchange 2007 never change the PidTagMessageClass property in this way.

<65> Section 3.1.4.6.2.2: Outlook 2003 and Outlook 2007 both copy the PidLidAppointmentAuxiliaryFlags to the Meeting object but Exchange 2003 and Exchange 2007 do not.

<66> Section 3.1.4.6.2.2: Outlook 2007, Exchange 2007, and Exchange 2003 set this property, but Outlook 2003 does not.

<67> Section 3.1.4.6.2.2: Outlook 2007 and Exchange 2007copy the RecipientRows of the PidLidAppointmentUnsendableRecipients property of the Meeting Request object to the RecipientRows of the Meeting object. Outlook 2003 and Exchange 2003 do not.

<68> Section 3.1.4.6.2.2: Outlook 2003 and Outlook 2007 do not copy the Busy status for the exception.

<69> Section 3.1.4.6.2.2: Outlook 2003 and Outlook 2007 both set PidTagProcessed. Exchange 2003 and Exchange 2007 do not set this flag.

<70> Section 3.1.4.6.2.2: Exchange 2007, Exchange 2003, Outlook 2007, and Outlook 2003 do not set this property.

<71> Section 3.1.4.6.2.2: Exchange 2007 sets this property, but Exchange 2003, Outlook 2007, and Outlook 2003 do not.

<72> Section 3.1.4.6.2.3: Outlook 2007 does not automatically send Meeting Response objects if this property is set, but Outlook 2003, Exchange 2007, and Exchange 2007 do.

<73> Section 3.1.4.6.2.3: Exchange 2007, Exchange 2003, Outlook 2007, and Outlook 2007 do not set this property.

<74> Section 3.1.4.6.3: Outlook 2007 and Exchange 2007 will set the "old" properties. Outlook 2003 and Exchange 2003 will not set these properties.

<75> Section 3.1.4.6.3.1: Outlook 2007 and Exchange 2007 will set the value of the PidLidMeetingType to mtgInfo in this case. Outlook 2003 and Exchange 2003 will set the value of this property to mtgFull.

<76> Section 3.1.4.6.3.2: Outlook 2003 and Exchange 2003 will always clear responses whenever any update is sent out.

<77> Section 3.1.4.6.3.2: Outlook 2003 and Outlook 2007 set the PidTagRecipientTrackStatusTime value to 12:18 A.M. 23 October 1602. Exchange 2003 and Exchange 2007 do not change this value. Changing this value is not required.

<78> Section 3.1.4.6.3.4: Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 all give the user a choice about whether they want to send the update to all recipients or only added/removed recipients.

<79> Section 3.1.4.6.3.4: Outlook 2007 will treat an attendee that has been marked sendable as a new attendee, but Outlook 2003, Exchange 2007, and Exchange 2003 do not.

<80> Section 3.1.4.6.3.4: Outlook 2007 and Exchange 2007 set the PidLidAppointmentUnsendableRecipients as described, while Outlook 2003 and Exchange 2003 do not.

<81> Section 3.1.4.6.3.5: Outlook 2007 does this, but Outlook 2003 and Exchange do not.

<82> Section 3.1.4.6.3.5: Outlook 2007 sends out cancelations to exceptions when the recurrence pattern has changed, but Outlook 2003 does not.

<83> Section 3.1.4.6.3.5: Outlook 2007 sends Meeting Request objects for exceptions when the organizer adds attendees to the series and sends a Meeting Update object to a Partial attendee List.

<84> Section 3.1.4.6.4: Outlook 2007 and Exchange 2007 support the calendar dictionary, but Outlook 2003 and Exchange 2003 do not.

<85> Section 3.1.4.6.4.1: Outlook 2007 skips automatic updating of the Meeting object based on the values of these properties, but Outlook 2003, Exchange 2007, and Exchange 2003 do not.

<86> Section 3.1.4.6.4.2: Outlook 2007 does not recreate the exception if these properties are set, but Outlook 2003, Exchange 2007, and Exchange 2003 do not.

<87> Section 3.1.4.6.4.2: Exchange 2007, Exchange 2003, Outlook 2007, and Outlook 2003 do not set this property.

<88> Section 3.1.4.6.4.2: Outlook 2007 copies these properties onto the Meeting Update object, while Outlook 2003, Exchange 2003, and Exchange 2007 do not.

<89> Section 3.1.4.6.4.2: Outlook 2007 does not perform these actions if these properties are set but Outlook 2003, Exchange 2007, and Exchange 2003 do.

<90> Section 3.1.4.6.4.2: Exchange 2007, Exchange 2003, Outlook 2007, and Outlook 2003 do not set this property.

<91> Section 3.1.4.6.4.2: Outlook 2007 does not overwrite a private value of PidTagSensitivity, but Outlook 2003, Exchange 2003, and Exchange 2007 do.

<92> Section 3.1.4.6.4.2: Outlook 2007 and Exchange 2007 allow a Meeting object to be updated without changing the value of the PidLidResponseStatus property. Outlook 2003 and Exchange 2003 reset the value of this property to respNotResponded.

<93> Section 3.1.4.6.4.2: Outlook 2003 and Outlook 2007 both set PidTagProcessed. Exchange 2003, and Exchange 2007 do not set this flag.

<94> Section 3.1.4.6.4.2: Exchange 2007, Exchange 2003, Outlook 2007, and Outlook 2003 do not set this property.

<95> Section 3.1.4.6.4.2: Exchange 2007 sets these properties, but Exchange 2003, Outlook 2007, and Outlook 2003 do not.

<96> Section 3.1.4.6.5: Outlook 2003 and Outlook 2007 set the value of the PidLidMeetingType property to 0x00000000.

<97> Section 3.1.4.6.5: Outlook 2007 and Exchange 2007 write the PidLidAppointmentUnsendableRecipients property, but Outlook 2003SP3 and Exchange 2003 do not.

<98> Section 3.1.4.6.5.1: Outlook 2007 forwards exceptions to a recurring series, but Outlook 2003 SP, Exchange 2007, and Exchange 2003 do not.

<99> Section 3.1.4.7.1: Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 allow the end user to decide whether or not the end user wants to send a response to the organizer.

<100> Section 3.1.4.7.4: Outlook 2003 and Exchange 2003 will allow an organizer to send a response to their own meeting, but only if the asfReceived bit is not set in the value of the PidLidAppointmentStateFlags property. Outlook 2007 and Exchange 2007 will not allow an organizer to respond to their own meeting.

<101> Section 3.1.4.7.4: .Often when the organizer sends a Meeting Request object to a very large set of people, the organizer does not want to be flooded with Meeting Response objects. Regardless of the reason, when the property is set, the client SHOULD NOT send Meeting Response objects for the meeting.

<102> Section 3.1.4.7.4: Outlook 2003 and Outlook 2007 also write the following properties, which are not used by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2007: PidLidInternetAccountName, PidLidInternetAccountStamp, PidLidSendMeetingAsIcal

<103> Section 3.1.4.7.4: Outlook 2003 and Outlook 2007 also write the following properties when the Meeting Response object represents a recurring series. These are not used by Outlook 2003, Outlook 2007, Exchange 2003, or Exchange 2007:PidLidRequiredAttendees, PidLidOptionalAttendees, PidLidResourceAttendees, PidLidDelegateMail, PidLidSingleInvite, PidLidTimeZone, PidLidStartRecurrenceDate, PidLidStartRecurrenceTime, PidLidEndRecurrenceDate, PidLidEndRecurrenceTime, PidLidDayInterval, PidLidWeekInterval, PidLidMonthInterval, PidLidYearInterval, PidLidMonthOfYearMask, PidLidRecurrenceType, PidLidAllAttendeesList

<104> Section 3.1.4.7.5: Outlook 2007 and Exchange 2007 support the calendar dictionary, but Outlook 2003 and Exchange 2003 do not.

<105> Section 3.1.4.7.5.1: Outlook 2003 will process the response regardless of the value of PidLidServerProcessingActions property.

<106> Section 3.1.4.7.5.1: When Exchange 2007 processes a Meeting Response object that represents a exception on a Recurring appointment, Exchange 2007 will not record the response but still sets the cpsUpdatedCalItem bit of the PidLidServerProcessingActions property to 1. In this case, Office ignores the *cpsUpdatedCalItem* bit and still records the response.

<107> Section 3.1.4.7.5.2: Outlook 2007 and Exchange do not set the PidLidPromptSendUpdate property.

<108> Section 3.1.4.7.5.2: Outlook 2003, Outlook 2007 and Exchange do not verify that the attendee exists on an out-of-date Meeting Response object.

<109> Section 3.1.4.7.5.2: Outlook 2003, Outlook 2007 and Exchange do not add the attendee as an optional attendee if the Meeting Response object is out-of-date.

<110> Section 3.1.4.7.5.2: Outlook 2003 and Outlook 2007 compare the two time values rounded down to the nearest minute so that if an attendee responds twice within the same minute, both responses will be seen as having been sent at the same time. Exchange 2003 and Exchange 2007 do not round the time value.

<111> Section 3.1.4.7.5.2: Outlook 2003 and Outlook 2007 round the time value from the PidLidAttendeeCriticalChange property down to the nearest minute before setting the value in the PidTagRecipientTrackStatusTime property. Exchange 2003 and Exchange 2007 do not round the time value.

<112> Section 3.1.4.7.5.2: Outlook 2003 and Outlook 2007 allow the user to decide whether to "Delete empty responses." Exchange 2003 and Exchange 2007 never automatically delete responses.

<113> Section 3.1.4.8.1.1: Outlook 2007 will send cancelations to attendees marked not sendable, but Outlook 2003, Exchange 2007, and Exchange 2003 will not.

<114> Section 3.1.4.8.1.2: Outlook 2003 sends out cancellations for deleted exceptions when sending out a cancellation for a recurring series but Outlook 2007, Exchange 2003, and Exchange 2007 do not.

<115> Section 3.1.4.8.1.2: Outlook 2007 sends meeting Cancelation objects to exceptions when sending a meeting Cancelation object to a recurring series to a Partial attendee List, but Outlook 2003 and Exchange do not.

<116> Section 3.1.4.8.2: Outlook 2007 and Exchange 2007 support the calendar dictionary, but Outlook 2003 and Exchange 2003 do not.

<117> Section 3.1.4.8.2.1: Outlook 2003 and Exchange do not skip automatic updating of the Meeting object based on these properties.

<118> Section 3.1.4.8.2.1: Outlook 2003 and Outlook 2007 will recreate the Exception object, but Exchange 2003 and Exchange 2007 will not.

<119> Section 3.1.4.8.2.1: Outlook 2003 and Outlook 2007 will create the Meeting object, but Exchange 2003 and Exchange 2007 will not.

<120> Section 3.1.4.8.2.2: Outlook 2003 and Outlook 2007 both set PidTagProcessed. Exchange 2003 and Exchange 2007 do not set this flag.

<121> Section 3.1.4.9.1: Exchange will send a meeting Forward Notification regardless of the value of the PidLidAppointmentStateFlags property.

<122> Section 3.1.4.9.1: Office also writes the following properties, which are not used by Office or Exchange : PidLidInternetAccountName, PidLidInternetAccountStamp, and PidLidSendMeetingAsIcal

<123> Section 3.1.4.9.1: Office also writes the following properties when the Meeting Response object represents a recurring series. These are not used by Office or Exchange:PidLidRequiredAttendees, PidLidOptionalAttendees, PidLidResourceAttendees, PidLidDelegateMail, PidLidSingleInvite, PidLidTimeZone, PidLidStartRecurrenceDate, PidLidStartRecurrenceTime, PidLidEndRecurrenceDate, PidLidEndRecurrenceTime, PidLidDayInterval, PidLidWeekInterval, PidLidMonthInterval, PidLidYearInterval, PidLidMonthOfYearMask, PidLidRecurrenceType, and PidLidAllAttendeesList

<124> Section 3.1.4.9.2: Outlook 2007 and Exchange 2007 support the calendar dictionary, but Outlook 2003 and Exchange 2003 do not.

<125> Section 3.1.5.4: Outlook 2007 and Exchange 2007 increment the sequence number when sending a Meeting Update object for an exception but Outlook 2003 and Exchange 2003 do not.

<126> Section 3.1.5.4: If the new sequence number is set in the PidLidAppointmentSequence property of the Meeting object when the Meeting Request object is only sent to Added/Removed attendees, any meeting responses from the original attendees will not be recorded on the Meeting object. Exchange 2007 does set the new sequence number in the PidLidAppointmentSequence property.

<127> Section 3.1.5.5.1: Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 do not interpret data in this manner.

<128> Section 3.1.5.5.2: Outlook 2003, Outlook 2007, Exchange 2003, and Exchange 2007 do not interpret data in this manner.

<129> Section 3.1.5.6: Outlook 2003, Exchange 2003, and Exchange 2007 do not support the PidTagScheduleInfoDelegatorWantsInfo property.

<130> Section 3.1.5.6: Office, Exchange 2003, and Exchange 2007 do not set the cpsDelegatorWantsCopy bit of the PidLidServerProcessingActions property.

<131> Section 4.2.2.3: Exchange 2010 can output unexpected results when using RopOpenMessage when client connection services are deployed on an Exchange server that does not also have a Mailbox store installed.

<132> Section 4.2.2.7: If a match had not been found, a client would search for an orphan instance by trying to match the value of the PidLidGlobalObjectId property from the Meeting Update object (because this Meeting Update object represents an exception). If an orphan instance is not found, a client would search for a matching row with the PidTagOwnerAppointmentId value of 0 (zero). If a matching recurring series or orphan exception still could not be found, then it would be assumed that the Meeting object does not exist in the folder and the Meeting Update object would be treated as a Meeting Request object.

<133> Section 4.2.2.7: If the Exception Attachment object has the PidTagExceptionReplaceTime property, the value of this property is compared with the computed Replace Time to determine if the attachment is the matching exception. If the attachment does not have this property, the client needs to use RopOpenAttachment, RopOpenEmbeddedMessage, and RopGetPropertiesSpecific to get the PidLidExceptionReplaceTime property from the Exception Embedded Message object, and match that value against the computed Replace Time.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index