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

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability.
Microsoft Corporation	April 25, 2008	0.2	Revised and updated property names and other technical content.
Microsoft Corporation	June 27, 2008	1.0	Initial Release.
Microsoft Corporation	August 6, 2008	1.01	Revised and edited technical content.

Table of Contents

1 Introduction

The concept of calendaring enables 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 being requested at an event, users can accept, decline, or propose a different date and/or time. Delegation enables one user to manage the schedule 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 format for storing events as calendar objects
- A process where a client or a server can retrieve these objects
- 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 the Glossary section of [MS-OXGLOS]:

Address Book object
Appointment
Appointment object
Attachment object
binary large object (BLOB)
Boolean
BYTE
Calendar Folder
Calendar object
Coordinated Universal Time (UTC)
delegate
Delegate Information object
Delegator
exception
DWORD
Embedded Message object
Exception Attachment Object
Exception Embedded Message Object
Exception object
GUID
Informational update
Meeting Object
meeting related object
Meeting Request Object

Meeting Response Object
Meeting Update Object
Meeting Workspace
property (1)
public folder
Recurring Calendar Object
Sent Mail folder
signal time
special folder
Task object
Unicode

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

ULONG
LONG
SYSTEMTIME

The following terms are specific to this document:

Attendee: A person who is invited to attend a meeting.

Calendar Special Folder: A **Calendar Folder** in a user's mailbox into which meetings will be created by default. See [MS-OXOSFLD].

Counter Proposal: A request from an **Attendee** to the **Organizer** to change the date and/or time of a **Meeting**

Full Update: A **Meeting Update Object** that includes a change to date and/or time or **Recurrence Pattern**, which requires a response from **Attendees**

Instance: A single occurrence of an **Appointment object** or **Meeting Object** that has a **Recurring Series** specified.

Meeting Cancelation Object: A **Message object** sent to **Attendees** when the **organizer** of a **meeting** cancels a previously scheduled event.

Meeting Request: An **Instance** of a **Meeting Request Object**.

Meeting Update: An **Instance** of a **Meeting Update Object**.

Optional Attendee: An **Attendee** of an event that 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**.

Organizer: The owner of a Meeting.

Recurring Series: An event that repeats, at specific intervals of time, according to a **Recurrence Pattern**.

Recurrence Pattern: Information for a repeating event, such as the start and end time, the number of occurrences and how occurrences are spaced (daily, weekly, monthly, etc).

Replace Time: The original start date and time of an **Instance**, according to the **Recurrence Pattern**, to be replaced by the start date and time of the **exception**.

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. The Attendee can be a required, optional or resource Attendee.

Sequence Number: The revision number of a **Meeting Object**. It's used to determine the most recent **Meeting Update** sent by the **Organizer**.

Series: Same as **Recurring Series**.

Significant Change: A change made by an **Organizer** to a **Meeting Object** that requires sending out a **Meeting Update Object**.

single instance: An **Appointment object**, **Meeting Object**, or **Task object** that occurs only once.

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

[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, "Office Exchange 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, "Office Exchange Protocols Master Property List Specification", June 2008.

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

1.2.2 Informative References

None.

1.3 Protocol Overview

The Appointment and Meeting Object protocol specifies:

- The **Message objects** 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 specified by the Appointment and Meeting Object protocol can be divided into two classes of objects:

1. **Calendar objects**, for example, objects that are created and reside in a **Calendar Folder**. The two calendar object types are **Appointment objects** and **Meeting Objects**.
2. **Meeting related objects**, for example, objects that relay Meeting Object information from **Organizer** to **Attendees** and vice versa. These include **Meeting Request Objects**, **Meeting Update Objects**, **Meeting Cancelation Objects**, and **Meeting Response 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 being 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 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 indicating 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** desires to make changes to a previously scheduled meeting, the Organizer sends a special type of **Meeting Request Object**, called 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 Cancelation Object

The **Organizer** sends a **Meeting Cancelation Object** to notify **Attendees** that a previously scheduled event will not take place.

1.4 Relationship to Other Protocols

The Appointment and Meeting Object Protocol extends the [MS-OXCMSG] protocol for use with calendaring objects and relies on [MS-OXOMSG] for message transport and delivery.

1.5 Prerequisites/Preconditions

The Appointment and Meeting Object protocol specification assumes that the client has previously acquired a handle to the object (**Appointment, meeting**, **Meeting Request**, **meeting response**, **Meeting Update**, **meeting cancelation**, or **exception attachment**, or **exception embedded message**) 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. See [MS-OXCPRPT], [MS-OXCMSG], [MS-OXCFOLD].

1.6 Applicability Statement

The Appointment and Meeting Object protocol is appropriate for clients and servers which manage user **Appointments** and meetings and their associated resources.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

This protocol provides no 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 as specified in [MS-OXCPRPT] and [MS-OXCMSG] as its primary transport mechanism.

2.2 Message Syntax

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

Clients operate on calendar and meeting related objects 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 below, calendar and meeting related objects MUST adhere to all **property** constraints specified in [MS-OXPROPS] and all property constraints specified in [MS-OXCMSG]. An object MAY contain other properties as specified in [MS-OXPROPS] but these properties have no 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 MUST return an error and ignore any request to change the value of this property.

2.2.1 Common Properties

Unless otherwise noted in the special handling section below, the objects specified in the Appointment and Meeting Object protocol MUST include the common properties as specified in [MS-OXCPRPT]. The objects MUST 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 properties common to all object types in the Appointment and Meeting Object protocol. Unless otherwise specified, the properties below MUST 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 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. Section 3.1.5.4 describes more about when and how a client increments the sequence number.

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 below.

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.

2.2.1.3 PidLidAppointmentAuxiliaryFlags

Type: PtypInteger32
Specifies a bit field that describes the auxiliary state of the object. This **property** is not required. Below 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 SHOULD<5> send a **Meeting Response Object** back to the **Organizer** when a response is chosen.

F (auxApptFlagForwarded, 0x00000004): This flag on a Meeting Request Object indicates that it was forwarded (including being forwarded by the Organizer), rather than being an invitation 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; MUST be 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**.

2.2.1.6 PidLidAppointmentEndWhole

Type: PtypTime
Specifies the end date and time for the event; MUST be 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**.

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.<6>

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. It MAY instead display the calendar object based on the value of the PidNameKeywords **property** as specified in [MS-OXCMSG]. When set, the value MUST be one of the following.

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 or not 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 the event is not an all-day event. The client or server MUST 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. Below 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 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 in the table below.

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 the Attendee has not yet responded. This value is on the Meeting Request, Meeting Update, and Meeting Cancelation Objects.

2.2.1.12 PidLidRecurring

Type: PtypBoolean
Specifies whether or not 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 or not 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**<7>, 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, it 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 attendee's display name. Separate entries MUST be 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 MUST be 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 MUST be 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 MUST be delimited by a semicolon followed by a space. <8> 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 MUST be delimited by a semicolon followed by a space. <9> 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 MUST be delimited by a semicolon followed by a space. <10> 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 PidLidNonSendableTo property at the same index. <11>

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 PidLidNonSendableCc property at the same index <12>.

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 PidLidNonSendableBcc property at the same index <13>.

2.2.1.25 PidLidAppointmentUnsendableRecipients

Type: PtypBinary
This **property** contains a list of **Unsendable Attendees**. This property is not required but SHOULD be set. <14> It has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
RowCount																															
RecipientRow[1..RowCount]																															

RowCount: The count of *RecipientRow*
RecipientRow: A list recipient of table rows. See [MS-OXOCMSG]. See also the additional properties in section 2.2.3.9 that can be set on RecipientRows for calendar 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/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/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. Once set for a calendar object, the value of this **property** MUST NOT change. The fields in this **BLOB** are specified below. All fields have little-endian byte ordering.

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: 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: The high-ordered byte of the 2-byte Year from the PidLidExceptionReplaceTime property if the object represents an **exception**, otherwise zero.

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

M: 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: The Day of the month from the PidLidExceptionReplaceTime property if the object represents an exception, otherwise zero.

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

X: Reserved, MUST be all zeroes.

Size: A LONG value defining the size of the Data component.

Data: An array of bytes that ensures 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 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 can assist a client or server in finding a calendar object but is not guaranteed to be unique among all objects.<15> This **property** is not required on objects.

2.2.1.30 PidTagStartDate

Type: PtypTime

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

2.2.1.31 PidTagEndDate

Type: PtypTime

For backward compatibility with older clients, this SHOULD be set, and when set, 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 MUST be 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** MUST NOT be sent to the **Organizer**. When the value of this property is TRUE, and the client or server automatically responds (see sections 2.2.10.2-2.2.10.4), a Meeting Response Object MUST be sent to the Organizer. Otherwise, when the value is TRUE, the client or server MAY<16> 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 or **Meeting related object**. The contents SHOULD use the RTF properties [MS-OXRTFCP] for objects specified by the Appointment and Meeting Object protocol. When stored and retrieved, **BestBody** guidance as specified in [MS-OXBBODY] MUST 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.

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
lBias																															
lStandardBias																															
lDaylightBias																															
wStandardYear																stStandardDate															
…																															
…																															
…																															
…																wDaylightYear															

stDaylightDate
…
…
…

lBias: The time zone's offset in minutes from UTC.

lStandardBias: The offset in minutes from lBias during Standard Time.

lDaylightBias: The offset in minutes from lBias during Daylight Time.

wStandardYear: This matches the stStandardDate's wYear member.

stStandardDate: SYSTEMTIME structure as specified in [MS-DTYP]. It contains the date and local time indicating when to begin using the lStandardBias.

If the time zone does not support daylight saving, 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).
wDaylightYear: this matches the stDaylightDate's wYear field.

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

2.2.1.40 PidLidTimeZoneDescription

Type: PtypString
Specifies a human-readable description of the time zone 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 represented by PidLidTimeZoneStruct, then the client MUST use PidLidTimeZoneStruct instead of this property<17>. If this property is not set, PidLidTimeZoneStruct will be used instead <18>. The fields in this BLOB are encoded in little-endian byte order.

										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Major Version								Minor Version								cbHeader															
Reserved																cchKeyName															
KeyName (variable)																															
cRules																TZRules [1..cRules]															

Major Version: This is set to 0x02.

Minor Version: This is set to 0x01.

cbHeader: The count of bytes contained in Reserved, cchKeyName, KeyName, and cRules.

Reserved: This Word field MUST be set to 0x0002:

cchKeyName: This WORD property represents the count of characters in the KeyName field that follows.

KeyName: **Unicode** string identifying the associated time zone. The string SHOULD NOT be localized and MUST be set to the unique name of the desired time zone <19>. This string has a maximum length of 260 characters, and it is not null terminated.

cRules: This WORD property represents the count of TZRules. Minimum is 1, maximum count is 1024.

TZRules: 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 MUST be 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. Below is 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 manner:

										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Major Version								Minor Version								Reserved															

TZRule Flags	wYear
X	
…	
…	
…	lBias
…	lStandardBias
…	lDaylightBias
…	stStandardDate
…	
…	
…	
…	stDaylightDate
…	
…	
…	
…	

Major Version: This is set to 0x02.

Minor Version: This is set to 0x01.

Reserved: This MUST be set to0x003E.

TZRule Flags: 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, then both of these flags are set (for example, the value is 0x0003). If this is not the active time zone rule, then neither of these flags are set. These flags are set on exactly one TZRule contained in this **property**, and all of the other rules MUST NOT have any flags set.

wYear: WORD representing 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, then this rule will remain in effect indefinitely.

X: Unused, MUST be all zeros.

lBias: LONG representing the time zone's offset in minutes from UTC.

lStandardBias: LONG representing the offset in minutes from lBias during Standard Time.

lDaylightBias: LONG representing the offset in minutes from lBias during Daylight Time.

stStandardDate: SYSTEMTIME structure as specified in [MS-DTYP]. It contains the date and local time indicating when to begin using the lStandardBias.

If the time zone does not support daylight saving, 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]. It contains the date and local time indicating when to begin using the lDaylightBias. This property has the same format and constraints as stStandardDate field above.

2.2.1.42 PidLidAppointmentTimeZoneDefinitionStartDisplay

Type: PtypBinary
Specifies time zone information that indicates the time zone of the PidLidAppointmentStartWhole **property**<20>. 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 2.2.1.41, with one **exception**. For each TZRule specified by this property, the R flag in the TZRule Flags MUST NOT be 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**<21>. 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 using one of the **Recurrence Patterns** and ranges specified below. The value of this **property** also contains information about both modified and deleted **exceptions**; information such as dates, subject, location, and several other properties of exceptions. The binary data in this property for **Recurring Calendar Objects** is stored as the *AppointmentRecurrencePattern* structure specified in section 2.2.1.44.2. This property MUST NOT exist on **single instance** calendar objects.

There are some limitations to recurrences:

- Multiple **Instances** MUST NOT start on the same day.
- Occurrences MUST NOT overlap – specifically, an exception that modifies the start date of an Instance in the Recurring Series MUST occur 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**.<22>

The schedule of a Recurring Series is determined by its Recurrence Pattern and range. This section describes the types of the recurrence range and Recurrence Patterns supported by this protocol.

Recurrence Range

The recurrence range identifies how long the event will continue. This protocol supports three different ranges: 1) ends after a specific number of occurrences 2) ends by a given date, 3) continues indefinitely.

Recurrence Pattern

The **Recurrence Pattern** determines the frequency of the event. The RecurrencePattern structure is also used to define recurring tasks as described in [MS-OXOTASK].

The following types of recurrences are supported by this protocol:

Daily Recurrence

A daily Recurrence Pattern schedules events that follow **one** of the patterns listed below:

- Every *n* number of days
- Every weekday

An example of a daily recurrence is as follows: An event that repeats every three days, starting Monday April 30, 2007 through Friday June 8, 2007.

Weekly Recurrence

A weekly Recurrence Pattern schedules events that follow the pattern listed below:

- Every *n* weeks on one or more particular days of the week

An example of a weekly recurrence is as follows: An event repeats every two weeks, on Tuesdays, starting on Monday April 30, 2007 and ending after five occurrences.

Monthly Recurrence

A monthly Recurrence Pattern can schedule events that follow **one** of the patterns listed below:

- 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 Tuesday of the month.

As an example of a monthly recurrence: An event that repeats on the fourth of every month, effective Monday April 30, 2007, without an end date.

Every N Months Recurrence

This Recurrence Pattern is a combination of the monthly and weekly patterns. An every *n* months pattern can schedule events to follow one of the patterns listed below:

- On the *m*th 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 example of an every *n* months recurrence is as follows: An event that occurs the last Thursday of every two months effective March 12, 2007, with an end date of December 31, 2007.

Month End Recurrence

A month end Recurrence Pattern can schedule events to repeat the last day of every n months. An example of a month end recurrence is as follows: An event that repeats on the last day of every month, effective Monday April 30, 2007, without an end date.

Yearly Recurrence

A yearly Recurrence Pattern can schedule events to follow one of the patterns listed below:

- On the *m*th day of the *n*th month, of every year
- On any day of the week on the first, second, third, fourth, or last week of the *n*th month, of every year

An example of a yearly recurrence is as follows: A birthday that occurs every June 22, and is an all-day event.

Note:

The Yearly Recurrence Pattern is based on a 12-month interval and therefore uses the monthly recurrence parameters to represent all the yearly recurrences.

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.

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

The following Recurrence Pattern types are valid.

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.
MonthNth	0x0003	The event has an every *n*th month pattern.
MonthEnd	0x0004	The event has a month end recurrence.
HjMonth	0x000A	The event has a monthly recurrence in the Hijri calendar. For this PatternType, the CalendarType MUST be set to 0x0000.
HjMonthNth	0x000B	The event has an every *n*th month pattern in the Hijri calendar. For this PatternType, the CalendarType MUST be set to 0x0000.

Name	Value	Description
HjMonthEnd	0x000C	The event has a month end recurrence in the Hijri calendar. For this PatternType, the CalendarType MUST be set to 0x0000.

CalendarType

The following values are acceptable for the calendar type:<23>

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

Name	Value	Description
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. Below is how the value of this field is computed, for each recurrence type.

Daily Recurrence<24>

For this recurrence type, the value of FirstDateTime field is numerical value of *StartDate* modulo *Period.*

Weekly Recurrence<25>

This value is calculated as follows:

1. Find the first *FirstDOW* before *StartDate*.
2. Calculate the number of minutes between midnight that day and midnight, January 1, 1601.
3. Compute the value of Period multiplied by 10080, which is the number of minutes in a week.
4. Take the value computed in step 2 modulo the value computed in step 3.

Monthly or Yearly Recurrence<26>

This value is calculated as follows:

1. Find the first day of the month of *StartDate.*
2. Determine MinimumDate. For Gregorian Calendars, this is midnight, Jan 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.
3. Calculate the number of calendar months between midnight of the days calculated in step 1 and 2 above.
4. Take that value modulo *Period*.
5. Add that number of months to the MinimumDate as determined in step 2.

6. 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 and the *MaximumRecurrenceInterval*, which is 999 days for daily recurrences, 99 weeks for weekly recurrences and 99 months for monthly recurrences. This value is further specified below, for each recurrence type.

Daily Recurrence

For daily recurrence, the period is stored as the minutes in whole number of days. For example, to define a recurrence that occurs every 2 days, the period would be 0x00000B40, which equates to 2880 minutes, or 2 days.

Weekly Recurrence

For 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

For monthly and yearly recurrence, the *Period* field is stored in months. If the recurrence is a yearly recurrence, *Period* MUST be set to 12.

SlidingFlag

This is only used for scheduling tasks; otherwise the value MUST be 0. For more information 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 for each pattern type is specified below:

Day Recurrence Pattern (PatternType is 0x0000)

In this case, *PatternTypeSpecific* has no value and is 0 bytes. In other words, PatternTypeSpecific is not included in the BLOB when PatternType is 0x0000.

Week Recurrence Pattern (PatternType is 0x0001)

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
	Sa	F	Th	W	Tu	M	Su																								

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.

Month, MonthEnd, HjMonth, or HjMonthEnd Recurrence Pattern (PatternType is 0x0002, 0x0004, 0x000A, or 0x000C, respectively)

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

Day

The day of the month on which the recurrence falls.

MonthNth or HjMonthNth Recurrence Pattern (PatternType is 0x0003 or 0x000B, respectively)

										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
	Sa	F	Th	W	Tu	M	Su																								
N																															

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

N

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

N Values

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 2 months, the two fields of PatternTypeSpecific field are set to 0x0000003E and 0x00000005.

If an event occurs on the first weekday of every 2 months, the two fields of PatternTypeSpecific field are set to 0x0000003E and 0x00000001.

If an event occurs on the last weekend day of every 1 month, the two fields of PatternTypeSpecific field are set to 0x00000041and 0x00000005.

If an event occurs on the first weekend day of every 1 month, the two fields of 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 table below.

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

OccurrenceCount

The number of occurrences in a recurrence.
When the End Type of the pattern is “End after date", this value MUST be computed. Although this field’s value MUST always be set, its value has no meaning on a Recurring Series with no end date.<27>

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
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 is the array of the original Instance date of deleted Instances. There is exactly one element for each deleted Instance and every deleted Instance MUST be represented in this array. Every modified Instance MUST also 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 MUST be equal to 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 MUST be ordered from earliest to latest. There SHOULD NOT<28> 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 is the array of the original Instance date of modified Instances. There is exactly one element for each modified Instance and every modified Instance MUST be represented in this array. Every modified Instance MUST also have an entry in the array of DeletedInstanceDates.
The count of the array MUST be equal to 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 MUST be ordered from earliest to latest. There SHOULD NOT<29> 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 MUST be calculated as the end date.
If the recurrence does not have an end date, *EndDate* MUST be set to 0x5AE980DF.

2.2.1.44.2AppointmentRecurrencePattern 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 is a *RecurrencePattern* structure that defines the recurrences. For detail, see 2.2.1.44.1

ReaderVersion2

This value MUST be set to 0x00003006.

WriterVersion2

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

StartTimeOffset

The number of minutes since midnight each occurrence starts on.
For example, the value for midnight is 0 and the value for 12 P.M. is 720.

EndTimeOffset

The number of minutes since midnight each occurrence ends on. For example, the value for midnight is 0 and the value for 12 P.M. is 720.

ExceptionCount

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

ExceptionInfo

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

StartDateTime	
EndDateTime	
OriginalStartDate	
OverrideFlags	SubjectLength*
SubjectLength2*	Subject(Variable)*
MeetingType*	
ReminderDelta*	
ReminderSet*	
LocationLength*	LocationLength2*
Location(Variable)*	
BusyStatus*	
Attachment*	
SubType*	
AppointmentColor*	

*The presence of this field is conditional upon the value of the *OverrideFlags* field. For more information, see the section below on the *OverrideFlags* field.

StartDateTime

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

EndDateTime

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

OriginalStartDate

The original starting time of the exception in local time in minutes since midnight, January 1, 1601.

OverrideFlags

A bit field that specifies what data is present in the *PropertyData* field, indicating that the exception has a different value than the **Recurring Series**. The table below summarizes the valid flags for this field.

Flag	Value	Comments
ARO_SUBJECT	0x0001	Indicates that Subject, SubjectLength and SubjectLength2 fields are present.
ARO_MEETINGTYPE	0x0002	Indicates that MeetingType field is present.
ARO_REMINDERDELTA	0x0004	Indicates that ReminderDelta field is present.
ARO_REMINDER	0x0008	Indicates that ReminderSet field is present.
ARO_LOCATION	0x0010	Indicates that Location, LocationLength, LocationLength2 are present.
ARO_BUSYSTATUS	0x0020	Indicates that BusyStatus is present.
ARO_ATTACHMENT	0x0040	Indicates that Attachment field is valid.
ARO_SUBTYPE	0x0080	Indicates that SubType is present.
ARO_APPTCOLOR<30>	0x0100	This flag is reserved and MUST NOT be set.
ARO_EXCEPTIONAL_BODY	0x0200	Indicates that the Exception Embedded Message Object has PidTagRtfCompressed property set on it. See 2.2.1.20.3 in [MS-OXCMSG] for more on PidTagRtfCompressed.

SubjectLength

The number of bytes of the *Subject* field plus 1.
This field is only present if the **ARO_SUBJECT** flag is set in the *OverrideFlags* field.

SubjectLength2

The number of bytes of the *Subject* field.
This field is only present if the **ARO_SUBJECT** flag is set in the *OverrideFlags* field.

Subject

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

The value of the **PidLidAppointmentStateFlags** property in the Exception Embedded Message Object. For possible values, see **PidLidAppointmentStateFlags** at 2.2.1.10. This field is only present if the **ARO_MEETINGTYPE** flag is set in the *OverrideFlags* field.

ReminderDelta

The value for the **PidLidReminderDelta** property (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

The value for the **PidLidReminderSet** property (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

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

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

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

The value for the **PidLidBusyStatus** property in the Exception Embedded Message Object. For possible values, see **PidLidBusyStatus**.This field is only present if the **ARO_BUSYSTATUS** flag is set in the *OverrideFlags* field.

Attachment

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

The value for the **PidLidAppointmentSubType** property in the Exception Embedded Message Object. For possible values, see **PidLidAppointmentSubType**.This field is only present if the **ARO_SUBTYPE** flag is set in the *OverrideFlags* field.

AppointmentColor

Reserved. This field MUST not be read from or written to.

ReservedBlock1Size

The size of the *ReservedBlock1* field. This MUST be 0.

ReservedBlock1

Reserved.

ExtendedException

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 upon the value of the *OverrideFlags* field. For more information, see the section on *OverrideFlags* in the *ExceptionInfo* structure above.

ChangeHighlight

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

										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
ChangeHighlightSize																															
ChangeHighlightValue																															
Reserved(Variable)																															

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.<31>

ReservedBlockEE1Size

The size of the *ReservedBlockEE1* field that follows. This MUST be 0.

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 starting 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 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 MUST be 0.

ReservedBlock2

Reserved.

2.2.1.45 PidLidRecurrenceType

Type: PtypInteger32
Specifies the recurrence type of the **Recurring Series** using one of the values listed below.

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 MUST be 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 of **Task objects** [MS-OXOTASK] related to the calendar object. This **property** is not required. <32>

2.2.1.48 PidLidMeetingWorkspaceUrl

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

2.2.1.49 PidTagIconIndex

Type: PtypInteger32
The value of this **property** indicates an icon used with the object. It SHOULD<33> be set to one of the following but MAY be -1.

Description	Value	Used by objects
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
Cancelation	0x00000408	Meeting Cancelation Object
Informational Update	0x00000409	Meeting Update Object

2.2.1.50 Deprecated properties

The following properties are deprecated and SHOULD NOT be written by clients or servers <34>. If PidLidConferencingCheck is set to FALSE, all the properties in this section are ignored. These properties MUST only 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" or "Windows Netmeeting" or "Exchange Conferencing". If this property is set, PidLidConferencingType MUST also be set. This property MUST be 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 following.

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 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 whose PidTagMessageClass property 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 base64 encoded value of PidLidGlobalObjectId property of the **Meeting Object**.
- Append the string "&p=" followed by the value of PidLidOnlinePassword property.
- Finally convert the string to **Unicode**.

If there are multiple Exchange Conferencing mailboxes in the BCC field, the value calculated using the last one 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 MUST be a maximum of 255 characters not including NULL.

2.2.2 Calendar Object

This section specifies properties that are specific to **calendar objects**. <35> Unless otherwise specified, these properties MUST 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<36> 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 Object**s 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 none of the Exception Embedded Message Objects has RecipientRows.<37>

2.2.3 Meeting Object

This section specifies the properties that are specific to **Meeting Objects**. These properties have no meaning for **Appointment objects**. <38> Unless otherwise specified, these properties MUST exist.

2.2.3.1 PidLidAppointmentSequenceTime

Type: PtypTime
The value of this **property** on the **organizer's Meeting Object** indicates the date and time at which the property PidLidAppointmentSequence was last modified. The value MUST be 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**. Section 3.1.5.4 describes more about when and how a client increments the sequence number. 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 MUST be specified in UTC.

2.2.3.4 PidLidFInvited

Type: PtypBoolean
This **property** indicates whether or not 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. Once 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 Object**. This property is only set 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 on PidTagMailboxOwnerName, see Store Object protocol specified in [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.<39> For more details on RecipientRow, see Message and Attachment Object protocol as specified in [MS-OXCMSG].

2.2.3.9 RecipientRow Properties

The **Meeting Object** MUST have one RecipientRow (as specified in [MS-OXCMSG]) for each **Sendable Attendee**. In addition, a RecipientRow MAY exist for the **Organizer** of the Meeting Object. **Unsendable Attendees** MUST 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 the following properties that can be set in the "Extra Properties" section of RecipientRows.

2.2.3.9.1 PidTagRecipientFlags

Type: PtypInteger32
Specifies a bit field that describes the recipient status. This **property** is not required. Below are the individual flags that can be set.

S (recipSendable, 0x00000001): The recipient is a **Sendable Attendee**. This flag is only used 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 only used 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 SHOULD be treated as if the corresponding recipient does not. This flag is only used in a RecipientRow of an Exception Embedded Message Object of the Organizer's Meeting Object.

X MUST NOT be set (reserved, 0x00000040) <40>

X MUST NOT be set (reserved, 0x00000080) <41>

G (recipOriginal, 0x00000100): Indicates the recipient is an original Attendee. This flag is only used in the PidLidAppointmentUnsendableRecipients property.

X (reserved, 0x00000200) <42>

2.2.3.9.2 PidTagRecipientTrackStatus

Type: PtypInteger32
The value of this **property** indicates the response status returned by the **Attendee**. If this value is not set, it MUST be assumed to be respNone. Otherwise, it MUST be one of the following as defined in the Response Table 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 MUST be 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 the most recent response from the Attendee did not include a new date/ time proposal. This value MUST 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 Object** or **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 from the Recipient Type Table MUST be set as the Recipient Type for each RecipientRow in the Meeting Object.

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 cancelation, and **Meeting Response 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 Requests** objects MUST 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 MUST be specified in UTC. <43>

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**. <44>

2.2.4.4 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, then the time zone referenced will not observe DST, otherwise the DST dates from the table below will be followed<45>.

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}
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}

Index	Standard offset from UTC+12 (international date line) in minutes	Standard date {wMonth, wDayOfWeek, wDay, wHour}	Daylight date {wMonth, wDayOfWeek, wDay, wHour}
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
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 MUST be interpreted as follows:

wMonth:

Value	Meaning
1	January

Value	Meaning
2	February
3	March
4	April
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 (12am) to 23 (11pm).

If DST 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**. <46> Unless otherwise specified, these properties MUST 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 indicating how the **Meeting Object** changed. <47> This **property** is not required. Below are the individual flags that can be set.

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

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

REC (BIT_CH_RECUR, 0x00000004): The **Recurrence Pattern** changed See the property PidLidAppointmentRecur.

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

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

AP (BIT_CH_ALLOWPROPOSE, 0x00000400): The property PidLidAppointmentNotAllowPropose 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. <48>

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 the following:

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 information, 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 information, 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**<49>. 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**<50>. 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**<51>. 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 Meeting Update Object for a Recurring Series MUST NOT include any **Exception Attachment Objects**. A separate Meeting Request Object or Meeting Update Object MUST 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 is the value of the CalendarType field from the PidLidAppointmentRecur **property**. Otherwise, this property is not set and MUST be assumed to be 0.

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 MAY 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 MUST be separated from the copied Meeting Object body with a delimiter according to the Delimiter Table, and then the delimiter MUST be followed by two blank lines. <52>

Delimiter Table

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

PidLidCalendarType	Delimiter
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 MUST 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** MUST be a localized string indicating accept, tentatively accept, or decline, unless the **Meeting Response Object** includes a new date/time proposal, in which case this MUST be indicated by the value of this property.<53>

2.2.6.3 PidLidAppointmentProposedStartWhole

Type: PtypTime
Specifies the proposed value for PidLidAppointmentStartWhole for a **Counter Proposal**. This value MUST be specified in UTC.

2.2.6.4 PidLidAppointmentProposedEndWhole

Type: PtypTime
Specifies the proposed value for PidLidAppointmentEndWhole for a **Counter Proposal**. This value MUST be 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 MUST be 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**.

2.2.7 Meeting Cancelation Object

This section specifies the properties that are specific to **Meeting Cancelation Objects**. Unless otherwise specified, these properties MUST 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** MUST be a localized string indicating that the meeting was canceled.<54>

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 Exceptions

An **exception** specifies changes to an **Instance** of a **Recurring Series**. There are two objects that define an exception: The **Exception Attachment Object** and the **Exception Embedded Message Object**. There SHOULD<55> be one Exception Attachment Object for each Instance listed in the ModifiedInstanceDates field of the PidLidAppointmentRecur **property** on the **calendar object**. There MUST be one Exception Embedded Message Object 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,** also 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 MUST exist.

2.2.8.1 Exception Attachment Object

The **Exception Attachment Object** MUST have the following properties:

2.2.8.1.1 PidTagAttachmentHidden

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

2.2.8.1.2 PidTagAttachmentFlags

Type: PtypInteger32
This **property** is specified in [MS-OXCMSG]. The value MUST include the afException (0x00000002) flag.

2.2.8.1.3 PidTagAttachMethod

Type: PtypInteger32
This **property** is specified in [MS-OXCMSG]. The value MUST be afEmbeddedMessage (0x00000005) indicating that the **exception** data in PidTagAttachDataObject is an **Embedded Message object**.

2.2.8.1.4 PidTagExceptionStartTime

Type: PtypTime
The value of this **property** indicates the start date and time of the **exception** in the local time zone of the machine when the exception is created. This property is informational and MUST NOT<56> be relied on for critical information.

2.2.8.1.5 PidTagExceptionEndTime

Type: PtypTime
The value of this **property** indicates the end date and time of the **exception** in the local time zone of the machine when the exception is created. This property is informational and MUST NOT<57> be relied on for critical information.

2.2.8.1.6 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 MUST be specified in UTC.<58>

2.2.8.2 Exception Embedded Message Object

The data stored in the **Embedded Message object** represented by the PidTagAttachDataObject **property** (see [MS-OXCMSG]) contains properties specific to the **exception**. Any property 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 MUST NOT be 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.

2.2.8.2.1 PidTagMessageClass

Type: PtypString8
The value of this **property** MUST be "IPM.OLE.CLASS.{00061055-0000-0000-C000-000000000046}".

2.2.8.2.2 Best Body Properties

If the value of the PidLidFExceptionalBody **property** is FALSE, body properties SHOULD NOT be written to the **Exception Embedded Message Object**. When body properties are written, they MUST follow the same rules as body properties for a calendar object.

2.2.8.2.3 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 MUST be in UTC.

2.2.8.2.4 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 MUST be in UTC.

2.2.8.2.5 PidLidExceptionReplaceTime

Type: PtypTime
This **property** specifies the date and time within the **Recurrence Pattern** that the **exception** will replace. The value MUST be specified in UTC. This property allows the **Exception Attachment Object** to be found for a particular **Instance**.

2.2.8.2.6 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, then the Exception Embedded Message Object MUST have a body. If the value of this property is FALSE, or if the property does not exist, then a client or server MUST obtain the body from the recurring calendar object.

2.2.8.2.7 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**, then 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.

2.2.9 Calendar Folder

In order 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<59> create them in the **Calendar Special Folder**.

2.2.9.1 PidTagContainerClass

Type: PtypString8
The value of this **property** for all **Calendar Folder**s MUST be set to “IPF.Appointment”.

2.2.9.2 PidTagDefaultPostMessageClass

Type: PtypString
If this **property** is set on a **Calendar Folder**, the value MUST contain either exactly “IPM.Appointment”, or begin with “IPM.Appointment.”.

2.2.10 Delegate Information Object

The following properties are set on the **Delegate Information object** specified in the Delegate Access Configuration Protocol. See [MS-OXODLGT] for more details.

2.2.10.1 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** as specific in the [MS-OXOPFFB] protocol. 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.

2.2.10.2 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. When responding, the response MUST be acceptance, unless for 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 MUST NOT automatically accept Meeting Requests. This is not a required property.

2.2.10.3 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 MUST decline **Meeting Request Objects** that represent a **Recurring Series**. A value of FALSE, or the absence of this property, indicates that recurring meetings MUST be accepted. This is not a required property.

2.2.10.4 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 MUST decline **Instances** that overlap previously scheduled events. A value of FALSE or the absence of this property indicates that overlapping Instances MUST be accepted. This is not a required property.

2.2.10.5 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:

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
Identifier																															
HeaderSize																															
Version																															
RecordsCount																															

RecordsSize
Records(Variable)[1...RecordsCount]

Identifier
This field MUST be the value 0xBEDEAFCD.

HeaderSize
This field MUST have the value 0x00000014.

Version
This field MUST have the value 0x00000003.

RecordsCount
The count of Records field.

RecordsSize
This field MUST have the value 0x00000014.

Records
Array of Record data structure where Record is defined as below:

Record

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
StartTime																															
EndTime																															
GlobalObjectIdSize																															
GlobalObjectId(Variable)																															
UsernameSize																Username(Variable)															

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

Objects specified in the Appointment and Meeting Object protocol extend the message object and has an abstract data model that does not differ from 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** MAY 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 MAY<60> create a clone of the meeting onto the Calendar Special Folder at the time of creation. All calendar objects MUST have all the required properties as specified under sections 2.2.1 and 0. A Meeting Object MUST also have the required properties as specified under 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 MUST set 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 MAY set the asfMeeting bit to 0, reverting the Meeting Object back to an Appointment object. However, once 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<61>

To copy a calendar object, the client MUST create a new calendar object in the target folder, and then copy all properties from the original object onto the new calendar object with the exception of the following properties.

- The following properties MUST NOT be copied onto the new object: PidLidAppointmentColor, PidLidGlobalObjectId, PidLidCleanGlobalObjectId, PidLidMeetingWorkspaceUrl.
- 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. <62>
- The auxApptFlagCopied bit MUST be set to 1 in the value of the PidLidAppointmentAuxiliaryFlags property on the new object.

3.1.4.3.1 Source Object is an Exception

When the source object is an **exception**, the client MUST create a new calendar object. The client MUST follow 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**, MUST 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 MUST copy 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 MUST create a new **Appointment object,** and after copying all properties from the source object, ensure that all required properties (according to sections 2.2.1 and 0) exist on the new Appointment object.

3.1.4.4 Deleting a Meeting Object

When the user deletes a **Meeting Object**, the client SHOULD<63> send a **Meeting Cancelation 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 StartDate and EndDate. The client MUST exclude every Instance that occurs on a DeletedInstanceDate and include every date in the ModifiedInstanceDate list. Note that the 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 MUST use the values from the StartDateTime and EndDateTime fields of the ExceptionInfo specified under section 2.2.1.44.2.

3.1.4.5.1 Finding an Exception

The AppointmentRecurrencePattern specified in section 2.2.1.44.1 specifies deleted **Instances** and modified Instances. Every modified Instance is associated with an **Exception Attachment Object** as specified in 2.2.8. For each modified Instance in the RecurrencePattern, there is a matching ExceptionInfo structure as specified under 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 corresponding 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** 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 MUST create 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 MUST modify the value of the PidLidAppointmentRecur **property** (as specified in section 2.2.1.44) in the following manner: The exception's new start date MUST be added to the ModifiedInstanceDate array. ModifiedInstanceCount MUST be incremented. The original start date MUST be added to the DeletedInstanceDate array and the DeletedInstanceCount MUST be incremented. The new and original start dates MUST be in the time zone specified by PidLidTimeZoneStruct. The ExceptionInfo as specified under section 2.2.1.44.2 MUST be 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 MUST also add an **Exception Attachment Object** and **Exception Embedded Message Object**, each with properties specified in section 2.2.8, and add any overridden properties to the Exception Embedded Message Object. The Exception Embedded Message Object's PidLidAppointmentStartWhole MUST be in **UTC** and MUST be the UTC equivalent of the date and time added to StartDateTime in the ExceptionInfo. The client MUST also copy 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 MUST be incremented and the date of the Instance being deleted MUST be added to the DeletedInstanceDate array.

3.1.4.5.4 Deleting an Exception

To delete an **exception**, the **Instance** being deleted MUST be removed from the ModificeInstanceDate array and the ModifiedInstanceCount MUST be decremented. The associated **Exception Attachment Object** MUST 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 MUST create and submit a new **Meeting Request Object**. The client MUST copy all properties specified in section 2.2.1 from the **Meeting Object** to the Meeting Request Object. The client also MUST add all required properties specified in section 2.2.5. The client MUST then set 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 according to section 2.2.1.13.
- The value of the PidLidRecurring property according to section 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 property PidTagProcessed MUST NOT be set.
- The value of the PidLidAllAttendeesString property, according to section 2.2.1.16.
- The value of the PidLidToAttendeesString property, according to section 2.2.1.17.
- The value of the PidLidCcAttendeesString property, according to section 2.2.1.18.
- The value of the PidTagStartDate property, according to section 2.2.1.30.
- The value of the PidTagEndDate property, according to section 2.2.1.31.

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), then 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 section 2.2.5.12.

After successfully sending a Meeting Request Object, the client MUST modify 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<64> 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 2.2.10.2).<65>
- The **Organizer** has permission to write to the attendee's **Calendar Special Folder** (see ACLs in [MS-OXCPERM]).

The client MUST fail the direct booking action and MUST 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 2.2.10.2).
- The value of the PidTagScheduleInfoDisallowOverlappingAppts property (see section 2.2.10.2) 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. To determine whether a conflict exists, see Section 3.1.4.12.

To Direct Book an attendee, the client MUST take the following actions:

- Create the Meeting Object on the attendee's Calendar Special Folder according to the specification in section 3.1.4.6.2.2, and then modify the Meeting Object as if the Attendee had accepted it, as specified in section 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 representing in the Attendee on the Organizer's **Meeting Object**1.5.
- Set the value of the PidTagRecipientTrackStatusTime property to the current date and time on the RecipientRow representing 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 representing the Attendee in the Organizer's Meeting Object.
- Remove the RecipientRow representing 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

Sometime after receiving a **Meeting Request Object**, the client MUST check whether or not the **Calendar Object** is eligible for update according to section 3.1.4.6.2.1 to determine whether or not to create a **Meeting Object** in the user's **Calendar Special Folder** with the information in the Meeting Request Object. If the client does determine that the Meeting Object needs to be created, it MUST do so according to section 3.1.4.6.2.2. If the PiAutoProcess value in the Calendar Options Dictionary [MS-OXOCFG] is set to 0, the client SHOULD NOT<66> 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 MAY have its own defined mechanism for allowing the user to decide whether or not **Meeting Objects** will be automatically created upon receipt of a Meeting Request Object.

If the client decides to create the Meeting Object, the client MUST create it according to the rules specified in the remainder of this section.

3.1.4.6.2.1 Deciding to Create a Meeting Object

When the **Delegator** receives a non-private<67> **Meeting Request Object**, and the value of the PidTagScheduleInfoDelegatorWantsInfo **property** on the Delegator's **Delegate Information object** is set to TRUE, the client SHOULD change the value of the PidLidMeetingType property on the Meeting Request Object to mtgDelegatorCopy, and SHOULD NOT<68> automatically create the **Meeting Object** on the calendar. Instead, the **delegate's** client SHOULD be the one to create the Meeting Object on the delegator's **Calendar Special Folder**.

If any one of the following conditions is met, then the client MUST 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 (specified in section 2.2.10.5), indicating that another user already declined the meeting.

3.1.4.6.2.2 Creating the Meeting Object

Before creating the **Meeting Object**, the client MUST try to Find the calendar object, according to section 3.1.5.1, and MUST NOT create a new Meeting Object if a match was found. After creating a Meeting Object, the client MUST copy all the properties specified in section 2.2.1 from the **Meeting Request Object** onto the Meeting Object. The client also MUST add all required properties specified in section 2.2.3. The client MAY<69> 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 MUST set 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<70> copy the value of the PidLidAppointmentAuxiliaryFlags property from the Meeting Request Object to the Meeting Object.
- The client SHOULD remove the downlevel text (see section 2.2.5.12) from the body.

If the Meeting Request Object represents a **Recurring Series** and the Meeting Object was created, the client MUST search the folder for **Orphan Instances** of the meeting by matching the PidLidCleanGlobalObjectId property with that of the new Meeting Object. The client MUST convert any Orphan Instances that are found into **exceptions**, and then delete the Orphan Instances.

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 MUST NOT be set. <71>

3.1.4.6.2.3 Auto Respond

After creating the Meeting Object, the client MAY automatically send a **Meeting Response Object** to the **Organizer** if the value of the **property** PidTagScheduleInfoAutoAcceptAppointments in the Organizer's **Delegate Information**

object is nonzero. When sending the Meeting Response Object, the client MUST do so as specified in section 3.1.4.7. If the client chooses to automatically respond to **Meeting Request Objects**, it MUST also adhere to the requirements of the PidTagScheduleInfoDisallowRecurringAppts and PidTagScheduleInfoDisallowOverlappingAppts properties, accepting or declining meetings as appropriate.

When the client is acting for the **delegate**, and the client supports sending automatic responses, it MUST use 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 **property** PidLidFInvited on the **Meeting Object**). To do so, the client MUST create and submit 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 in the Meeting Object, the client SHOULD set the old value as the value of the PidLidOldLocation property on the Meeting Update Object. Similarly, if the value of the PidLidAppointmentStartWhole and/or PidLidAppointmentEndWhole properties were modified by the user in the Meeting Object, the client SHOULD set the old values as the value of the PidLidOldWhenStartWhole and PidLidOldWhenEndWhole properties, respectively.<72>

The client MUST modify 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**. When used within this section of the document, a significant change to a Meeting Object includes any of the following conditions:

- The value of the **property** PidLidAppointmentStartWhole changed.
- The value of the property PidLidAppointmentEndWhole 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<73> 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 **property** PidTagResponseRequested), and a significant change (according to 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.<74>

To clear the tallied responses, the client MUST set 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 MAY set the value of the PidTagRecipientTrackStatusTime property in each RecipientRow to an invalid date<75>.

3.1.4.6.3.3 Partial Attendee List

When a significant change (according to section 3.1.4.6.3.1) has not been made, and the user added **Attendees**, the client MAY<76> send the Meeting Update Object to only the new Attendees. In this case, the client SHOULD<77> add all other Attendees (for example, those not receiving the Meeting Update Object) into the PidLidAppointmentUnsendableRecipients **property** on the Meeting Update Object.

3.1.4.6.3.4 Updating a Recurring Series

After a **Meeting Update Object** is sent for a **Recurring Series** that has **exceptions**, the client MUST send a Meeting Update 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 Update Object for each exception MUST conform to the specifications in section 2.2.5.

3.1.4.6.4 Receiving a Meeting Update

Sometime after receiving a **Meeting Update Object**, the client determines, according to section 3.1.4.6.4.1, whether or not to update the **Meeting Object** in the user's **Calendar Special Folder** with the information in the Meeting Update Object. If the client decides that the Meeting Object needs to be updated, it MUST do so according to section 3.1.4.6.4.2. If the PiAutoProcess value in the Calendar Options Dictionary (see [MS-OXOCFG]) is set to 0, the client SHOULD NOT<78> immediately update the Meeting Object, but wait until the user views the Meeting Update Object. A client that does not support the Calendar Options Dictionary MAY have its own defined mechanism for allowing the user to decide whether or not **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 non-private<79> **Meeting Update Object**, and the value of the PidTagScheduleInfoDelegatorWantsInfo **property** on the Delegator's **Delegate Information object** is set to TRUE, the client SHOULD change the value of the PidLidMeetingType property on the **Meeting Request Object** to mtgDelegatorCopy, and SHOULD NOT<80> automatically update the **Meeting Object** in the **Calendar Special Folder**. Instead, the

delegate's client SHOULD be the one to update the Meeting Object in the delegator's Calendar Special Folder.

If any one of the following conditions is met, then the client MUST 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 section 2.2.10.5), indicating that another user already declined the meeting.

3.1.4.6.4.2 Updating the Meeting Object

When the client has determined that the **Meeting Object** is eligible for update, it MUST first try to find the calendar object, according to 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 MUST recreate the exception as specified in section 3.1.4.5.2. If the Meeting Object was not found, the client SHOULD change the value of the PidLidMeetingType **property** on the Meeting Update Object to mtgRequest, and then MUST follow the specification for receiving a new **Meeting Request Object** under section 3.1.4.6.2.

If the Meeting Update Object is out of date, as defined by section 3.1.5.2, then the client SHOULD change the value of the PidLidMeetingType property on the Meeting Update Object to mtgOutofDate and MUST NOT update the Meeting Object. Similarly, if the Meeting Update Object is not newer than the Meeting Object, as defined by section 3.1.5.3, the client MUST NOT update the Meeting Object.

Before modifying the Meeting Object, the client SHOULD<81> 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.

To update the meeting, the client MUST copy all the properties specified in section 2.2.1 from the Meeting Update Object onto the Meeting Object. The client also MUST add 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, then it MUST 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 MUST 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 (according to section 3.1.4.6.3.1), and the **Attendee** had already responded to the original Meeting Request Object, then the client SHOULD NOT<82> change the value of the PidLidResponseStatus property on the Meeting Object. On the other hand, regardless of whether or not the Attendee had previously responded, if the Meeting Update Object represents an update with a significant change (according to section 3.1.4.6.3.1), the client MUST set 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 MUST be set to olFree.

The client MUST follow the same rules surrounding Auto Respond for a Meeting Update Object as explained 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 MUST NOT be set. <83>

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 MUST create a new Meeting Request Object and copy all the properties from the original Meeting Request Object onto the new object. The client MUST then make 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 MUST be 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.
- The value of the PidLidChangeHighlight property to 0x00000000.
- The value of the PidLidMeetingType property to 0x00000000.
- Set the auxApptFlagForwarded bit to 1 in the PidLidAppointmentAuxiliaryFlags property.
- The client SHOULD copy all the RecipientRows from the original Meeting Request Object into the PidLidAppointmentUnsendableRecipients<84> 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 MAY<85> set the auxApptFlagForceMtgResponse bit in the PidLidAppointmentAuxiliaryFlags property.
- The property PidTagProcessed MUST NOT be set.

When a Meeting Request Object is forwarded, the client SHOULD<86> attempt to add a RecipientRow for the new Attendee to the **Meeting Object** in the Organizer's **Calendar Special Folder**, so the Organizer can see the full Attendee list.

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 MUST ensure that the **Meeting Object** has been created in the attendee's **Calendar Special Folder** according to section 3.1.4.6.2.2. Similarly, when the Attendee or delegate of the Attendee accepts a **Meeting Update Object**, the client MUST ensure that the Meeting Object has been updated in the attendee's Calendar Special Folder according to section 3.1.4.6.4.2, unless the Meeting Update Object is out of date according to 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, all changes made to the Meeting Object in the attendee's Calendar Special Folder SHOULD be atomic, for example, by creating a copy of the object, applying the changes, then overwriting the original Meeting Object.<87>

- 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 it's not the delegate who is responding, then PidLidAppointmentReplyName property is not set.

The client MAY<88> send a Meeting Response Object back to the **Organizer**, as specified in section 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 MUST follow the specification in section 3.1.4.7.1, except that when updating the Meeting Object, the following substitutions MUST be 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 MUST ensure that the Meeting Object has been created in the attendee's **Calendar Special Folder** according to section 3.1.4.6.2.2. Similarly, when the Attendee or delegate of the Attendee declines a **Meeting Update Object**, the client MUST ensure that the Meeting Object has been updated in the attendee's Calendar Special Folder according to section 3.1.4.6.4.2, unless the Meeting Update Object is out of date according to 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 MUST apply 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, then 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 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 it's not the delegate who is responding, then PidLidAppointmentReplyName property is not set.
- If it is the delegate acting on behalf of the **Delegator**, then 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 MUST remove Meeting Object from the attendee's calendar, either by moving the Meeting Object to the deleted objects 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 MUST remove the **Exception Attachment Object** from the Recurring Series, as specified in section 3.1.4.5.4.
- If it is the delegator or a delegate acting on behalf of the delegator, then a tombstone SHOULD be added to the PidTagScheduleInfoAppointmentTombstone property on delegator's **Delegate Information object**, as specified in section 2.2.10.5.

The client MAY send a **Meeting Response Object** back to the **Organizer**, as specified in section 3.1.4.7.4.

3.1.4.7.4 Sending a Meeting Response

After choosing a response, an **Attendee** or **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.<89>
- The value of the PidTagResponseRequested **property** on the **Meeting Request Object** is set to FALSE.<90>

If the following condition is true, the client SHOULD NOT allow the Attendee to choose a response *without* 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**).<91>

Outside the constraints above, the client MAY send a Meeting Response Object to the Organizer to inform them of the attendee's choice. To do so, the client MUST create and submit a new **Meeting Response Object**. The client MUST copy the following properties from the **Meeting Object** to the Meeting Response Object.<92>

- PidLidLocation
- PidLidWhere

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

In addition to the above, if the Meeting Response Object represents a **Recurring Series**, the client MUST copy the following properties from the Meeting Object.<93>

- 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 according to section 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 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 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 MUST set the following properties on the **Meeting Response Object**.

- The value of the PidTagSubjectPrefix property according to 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 above, 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

Sometime after receiving a **Meeting Response Object**, the client MUST decide, according to section 3.1.4.7.5.1, whether or not 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 MUST do so according to section 3.1.4.7.5.2. If the PiAutoProcess value in the Calendar Options Dictionary (see [MS-OXOCFG]) is set to 0, the client SHOULD NOT<94> immediately record the response, but wait until the user views the Meeting Response Object. A client that does not support the Calendar Options Dictionary MAY have its own defined mechanism for allowing the user to decide whether or not 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, then the client MUST 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.

3.1.4.7.5.2 Recording the Response

As long as the client has decided to record the response on the **Meeting Object**, it MUST find the calendar object, according to 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, then the client SHOULD NOT recreate the Exception Attachment Object on the Organizer's Meeting Object just to record the response. Instead, the response SHOULD be discarded. <95>
- If the Instance exists in 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.

If the Meeting Response Object is found to be out of date, according to section 3.1.5.2, then the response MUST NOT be recorded. Otherwise, the client needs to find the RecipientRow corresponding to the **Attendee** in the Organizer's Meeting Object. If the client cannot find a RecipientRow for the attendee, it MUST add a RecipientRow for the Attendee as an **Optional Attendee**. On the other hand, if a RecipientRow for the Attendee already existed, and the value of the PidTagRecipientTrackStatusTime **property** from the RecipientRow is a time later than the value of the PidLidAttendeeCriticalChange property on the Meeting Response Object, then the response from the Meeting Response Object MUST NOT be recorded. <96>

To record the response the client MUST set the following properties 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.<97>
- Set the recipExceptionalResponse bit to 1 in the PidTagRecipientFlags property if the Meeting Response Object represents an exception to a Recurring Series.

Whether or not the Meeting Response Object includes a new date/time proposal, additional properties MAY need to be set, see the next section for a discussion on new date/time proposals. After recording the response, the client MAY<98> delete the response if the value of 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 MUST take 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 with a 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 that 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, then the client MUST take 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 Cancelations

3.1.4.8.1 Sending a Meeting Cancelation

The **Organizer** or **delegate** of the Organizer sends a **Meeting Cancelation Object** to inform **Attendees** that they no longer need to attend the event. To do so, the client MUST create and submit a new Meeting Cancelation Object. The client MUST copy all properties from the **Meeting Object** to the Meeting Cancelation Object, with the exception/addition of those specified in section 2.2.7.

The client MUST modify the sequence number as specified in section 3.1.5.4.

The client MUST set the following on the Meeting Cancelation Object:

- Set all the bits in the value of the PidLidAppointmentStateFlags **property** that are set in this value on the Meeting Object, and then also set 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 according to section 2.2.7.2.

The following optional properties MUST also be set on the Meeting Cancelation Object:

- PidLidAllAttendeesString, according to section 2.2.1.16.
- PidLidToAttendeesString, according to section 2.2.1.17.
- PidLidCcAttendeesString, according to section 2.2.1.18.
- PidTagStartDate, according to section 2.2.1.30.
- PidTagEndDate, according to 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), then the value of this property SHOULD be set to the LONG value 0x5AE980E1.

After successfully sending a Meeting Cancelation Object, the client MUST modify the **Meeting Object** in the Organizer's **Calendar Folder** in the following ways:

- Set the value of the PidLidToAttendeesString property equal to the value that was set on the Meeting Cancelation Object
- Set the value of the PidLidCcAttendeesString property equal to the value that was set on the Meeting Cancelation 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 MUST send a **Meeting Cancelation Object** to the Attendees that were removed and MUST NOT send a Meeting Cancelation Object to any other Attendees.

3.1.4.8.2 Receiving a Meeting Cancelation

Sometime after receiving a **Meeting Cancelation Object**, the client MUST decide, according to section 3.1.4.8.2.1, whether or not to update the **Meeting Object** in the user's **Calendar Special Folder** with the information in the Meeting Cancelation Object. If the client decides that the Meeting Object needs to be updated, it MUST do so according to section 3.1.4.8.2.2. If the PiAutoProcess value in the Calendar Options Dictionary (see [MS-OXOCFG]) is set to 0, the client SHOULD NOT<99> immediately update the Meeting Object, but wait until the user views the Meeting Cancelation Object. A client that does not support the Calendar Options Dictionary MAY have its own defined mechanism for allowing the user to decide whether or not **Meeting Objects** will be automatically updated upon receipt of a Meeting Cancelation Object.

3.1.4.8.2.1 Deciding to Update a Meeting Object

If any one of the following conditions is met, then the client MUST NOT automatically update the **Meeting Object**:

- The **Meeting Cancelation 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 Cancelation Object is set to TRUE.

As long as the client has decided to update the Meeting Object, it MUST first try to find the calendar object, according to section 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, then the client SHOULD NOT<100> recreate the **Exception Attachment Object** and **Exception Embedded Message Object** on the recurring Meeting Object. If the Meeting Object was not found at all, the client SHOULD NOT<101> recreate it.

If the Meeting Update Object is out of date, as defined by section 3.1.5.2, then the client SHOULD change the value of the PidLidMeetingType property on the Meeting Update Object to mtgOutofDate and MUST NOT update the Meeting Object. Similarly, if the Meeting Cancelation Object is not newer than the Meeting Object, as defined by section 3.1.5.3, the client MUST NOT update the Meeting Object.

3.1.4.8.2.2 Updating the Meeting Object

To update the **Meeting Object**, the client MUST copy all the properties specified in section 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 MUST NOT be set. <102>

3.1.4.9 Determining Meeting Conflicts

To determine if a meeting conflicts with another, a client MAY 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 whose end date/time is greater than or equal to the start date/time of the given meeting *and* the start date/time is lesser than or equal to the end date/time of the given meeting is considered to be in conflict.

- Expand any recurring meetings. For instructions on how to do this, please see section 3.1.4.4. If multiple **Instances** or **exceptions** fall into the range, each of them MUST be considered as a **single instance** meeting for the purpose of this algorithm.

- If the size of the list is greater than or equal to one, 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. When so doing, the client MUST search in the **Calendar Special Folder** of the mailbox for whom the event was intended. This is typically the mailbox of the user that is logged on, but for the **delegate**, the client MUST search the **delegator's** folder for objects received on behalf of the delegator.

To look for the object, the client MUST first look for a calendar object whose PidLidGlobalObjectId **property** matches the value of the PidLidCleanGlobalObjectId property of the meeting-related object.

If the action is being applied to an **exception** of a **Recurring Series**, additional operations are required, depending on whether or not a matching Recurring Series object was found:

- Found. If a Recurring Series object was found, the client MUST attempt to find the **Exception Attachment Object** within 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 that the Exception Attachment Object cannot be found, a new one can be created.
- Not Found. If the Recurring Series object was not found, the client MUST look for a Recurring Series object whose PidLidGlobalObjectId property matches the value of the PidLidGlobalObjectId property of the meeting-related object. This would be the case, for **Instance**, if a user has been invited only to an **exception** of 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** responded to an older Meeting Request Object or Meeting Update Object instead of the most current Meeting Update Object. This section describes how the client can determine if the Meeting Request Object or Meeting Response Object is out of date. If one of the following conditions holds true, then the Meeting Request Object or Meeting Response Object MUST be considered 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 in the following table:

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 on the **Organizer's** Meeting Object that represents the Attendee.

3.1.5.3 Newer Meetings

A Meeting Request Object or Meeting Cancelation Object MUST be 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 holds true:

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

3.1.5.4 Incrementing the Sequence Number

When sending a **Meeting Update Object or Meeting Cancelation Object** for an **exception** of a **Recurring Series**, the sequence number MUST NOT be incremented. In this case, the client MUST set the value of the PidLidAppointmentSequence **property** on the Meeting Update Object or Meeting Cancelation Object equal to the value of the PidLidAppointmentLastSequence property from the **Meeting Object**.

When the object does not represent an exception of a Recurring Series, the sequence number set on the Meeting Update Object or Meeting Cancelation Object MUST be greater than the sequence number that was set on any previous Meeting Request Object, Meeting Cancelation Object, or Meeting Update Object. The client MUST get the value of the PidLidAppointmentLastSequence property from the Meeting Object and increment the value by 1, resulting in the new sequence number. The client MUST set 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 Meeting Cancelation Object.

If the Meeting Update Object or Meeting Cancelation Object is not being sent to all **Attendees** of the meeting, then the client SHOULD NOT<103> set this new sequence number as the value of the PidLidAppointmentSequence property of the Meeting Object. But when it is being sent to all Attendees, the client MUST set the new sequence number as the value of the PidLidAppointmentSequence property on the Meeting Object.

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 Samples

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

4.1.1.1 Sample Recurrence BLOB Without Exceptions

The following sample contains the binary recurrence data for the following appointment:

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

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

```
043004300B2001000000C02100000100000000000000320000002220000000C0000000000000
000000000000000008020BC0C20ADBC0C06300000093000005802000076020000000000000000
0000000000
```

The following table describes the content of the sample recurrence BLOB.

Name	Type	Size	Sample	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).

Name	Type	Size	Sample	Description
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)
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).

Name	Type	Size	Sample	Description
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	
WriterVersion2	**ULONG**	4	08 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 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.
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 Sample Weekly Recurrence BLOB with Exceptions

The following sample contains the binary recurrence data for the following **Meeting Request**. The Meeting Request is the same as the request that is used in 4.1.1.1, but in this example, the following information has been changed:

- 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 AM to Monday 4/16/2007 11:30 A.M.

The following is the recurrence BLOB:

```
043004300B2001000000C021000001000000000000032000000222000000C000000000
0000001000000A096BC0C01000000A096BC0C8020BC0C20ADBC0C063000000930000058
0200007602000001003499BC0C5299BC0CF898BC0C11002200210053696D706C6520526
563757272656E636520776974682065786365707469F6E730800070033342F34313431
0000000004000000000000000000000003499BC0C5299BC0CF898BC0C2100530069006D0
070006C006500200052006500630075007200720065006E00630065002000770069007 4
00680020006500780063006500700074069006F006E0073000700330034002F0034003
1003400310000000000000000000
```

Size: 0x0106 bytes

The following table describes the content of the modified sample recurrence BLOB.

Name	Type	Size	Sample	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).

Name	Type	Size	Sample	Description
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)
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)

Name	Type	Size	Sample	Description
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.
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 A.M.

Name	Type	Size	Sample	Description
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.

Name	Type	Size	Sample	Description
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 70 74 69 6F 6E 73	“Simple Recurrence with exceptions.”
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.

Name	Type	Size	Sample	Description
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 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	The modified Unicode subject is: "Simple recurrence with exceptions"
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 Sample Daily Recurrence BLOB with Exceptions

The following sample contains the binary recurrence data for the following appointment:

- Occurs every 3 days effective 4/7/2011 until 5/4/2011 from 8:00 AM to 8:30 AM.
- The **Instances** on 4/19/2011 and 4/22/2011 were deleted.

The following is the recurrence BLOB:

```
043004300A2000000000A0050000E010000000000000212000000A00000000000000020
00000A0C1DC0C80D2DC0C00000000207EDC0C0016DD0C06300000093000000E0010000FE
01000000000000000000000000
```

Size: 0x0054 bytes

The following table describes the content of the modified sample recurrence BLOB.

Name	Type	Size	Sample	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	

Name	Type	Size	Sample	Description
StartTimeOffset	**ULONG**	4	E0 01 00 00	The appointment's start time is 480 minutes past midnight or 8:00AM.
EndTimeOffset	**ULONG**	4	FE 01 00 00	The appointment's end time is 510 minutes past midnight or 8:30AM.
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 Sample N-Monthly Recurrence BLOB with Exceptions

The following sample contains the binary recurrence data for the following appointment:

- Occurs every third weekend day every 3 months starting 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:

```
043004300C200300000060AE000003000000000000000410000000300000022000000A00000
00000000002000000 6028C50C4028C70C02000000002EC50C4028C70C8028C30C6027D50C06
30000009300000480300000FC0300000200483 1C50CFC31C50CA82BC50C0000882BC70C3C2CC
70C882BC70C10000D000C006E6577206C6F636174696F6E00000000004000000000000000000
0000040000000000000000000000000882BC70C3C2CC70C882BC70C0C006E006500770020006C0
06F0063006100740069006F006E000000000000000000
```

Size: 0x00D2 bytes

The following table describes the content of the modified sample recurrence BLOB.

Name	Type	Size	Sample	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).

Name	Type	Size	Sample	Description
FirstDateTime	**ULONG**	4	60 AE 00 00	1. Find the first day of the month of the month of *StartDate:* 2/1/2008 2. 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 of 1601: 4885 3. Take that value modulo *Period*: 1 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. 2/1/1601 5. 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 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 3rd occurrence of these days (0x00000003).

Name	Type	Size	Sample	Description
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
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 2:00PM.
EndTimeOffset	**ULONG**	4	FC 03 00 00	The appointment's end time is 1020 minutes past midnight or 5:00PM.
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 2:00PM.
EndDateTime	**ULONG**	4	FC 31 C5 0C	The end time of the exception is 5/11/2008 5:00PM.
OriginalStartTime	**ULONG**	4	A8 2B C5 0C	The original start date and time of the occurrence was 5/10/2008 2:00PM.

Name	Type	Size	Sample	Description
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 2:00PM.
EndDateTime	**ULONG**	4	3C 2C C7 0C	The end date and time of the exception is 8/9/2008 5:00PM.
OriginalStartTime	**ULONG**	4	88 2B C7 0C	The original start date and time of the occurrence was 8/9/2008 2:00PM.
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:				
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.

Name	Type	Size	Sample	Description
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 2:00PM.
EndDateTime	**ULONG**	4	3C 2C C7 0C	The end date and time of the exception is 8/9/2008 5:00PM.
OriginalStartTime	**ULONG**	4	88 2B C7 0C	The original start date and time of the occurrence was 8/9/2008 2:00PM.
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 Sample Yearly Recurrence BLOB with Exceptions

The following sample contains the binary recurrence data for the following appointment:

- Occurs every April 19 effective 4/19/2011 from 8:00 AM to 8:30 AM.
- Move the **Instance** on 4/19/2012 to 4/21/2012.

The following is the recurrence BLOB for this recurrence:

```
043004300D200200000040FA01000C00000000000000001300000023200000 0A0000000000000
00100000060CCE40C01000000A0D7E40CA0C1DC0CDF80E95A063000000093000000E0010000FE
010000010080D9E40C9ED9E40C40CEE40C000000000000040000000000000000000000000000
000
```

Size: 0x0072 bytes

The following table describes the content of the modified sample recurrence BLOB.

Name	Type	Size	Sample	Description
ReaderVersion	**WORD**	2	04 30	
WriterVersion	**WORD**	2	04 30	
RecurFrequency	**WORD**	2	0D 20	The pattern of the recurrence is yearly.

Name	Type	Size	Sample	Description
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	6. Find the first day of the month of the month of *StartDate:* 4/1/2011 7. 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 of 1601: 4/1/2011-1/1/1601 is 4887 months 8. Take that value modulo *Period*: 4887 % 12 = 3 9. 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. 10. 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.

Name	Type	Size	Sample	Description
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	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 8:00AM.
EndTimeOffset	**ULONG**	4	FE 01 00 00	The appointment's end time is 510 minutes past midnight or 8:30AM.
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 8:00AM.
EndDateTime	**ULONG**	4	9E D9 E4 0C	The end date and time of the exception is 4/21/2012 8:30AM.
OriginalStartTime	**ULONG**	4	40 CE E4 0C	The original start date and time of the occurrence was 4/19/2012 8:00AM
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:				

Name	Type	Size	Sample	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.
ReservedBlock2Size	**ULONG**	4	00 00 00 00	No data in this skip block.

4.1.1.6 Sample Yearly Hebrew Lunar Recurrence BLOB with Exceptions

The following sample contains the binary recurrence data for the following appointment:

- Occurs every year on ניסן ג starting ot MA 00:8 morf ג' ניסן תשס"ח8:30 AM.
- 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:

```
043004300D20020008008075OA000C00000000000000003000000232000000A0000000000000
001000000207EDC0C01000000207EDC0C6074C40CDF80E95A0630000009300000E0010000FE
01000001000080DC0C1E80DC0C0080DC0C24023C0000000100000000000000004000000000000
000000000000000000000
```

Size: 0x007A bytes
The following table describes the content of the modified sample recurrence BLOB.

Name	Type	Size	Sample	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).

Name	Type	Size	Sample	Description
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 that midnight 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 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 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 3rd day of the month (in the Hebrew Lunar calendar)

Name	Type	Size	Sample	Description
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 8:00AM.
EndTimeOffset	**ULONG**	4	FE 01 00 00	The appointment's end time is 510 minutes past midnight or 8:30AM.
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 8:00AM.
EndDateTime	**ULONG**	4	1E 80 DC 0C	The end date and time of the exception is 4/7/2011 8:30AM.
OriginalStartTime	**ULONG**	4	00 80 DC 0C	The original start date and time of the occurrence was 4/7/2011 8:00AM

Name	Type	Size	Sample	Description
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 Samples

Included in this section is a sample of the **PidLidGlobalObjectId** and PidLidCleanGlobalObjectId BLOB properties that refer to an **exception** of a **Recurring Series**. As shown below, the data for the fields of the Global Obj ID BLOB are stored in little-endian byte ordering, unless otherwise specified.

4.1.2.1 PidLidGlobalObjectId

The following is the value of the PidLidGlobalObjectId **property** for an object that represents an **exception** of a **Recurring Series**. The **Instance** represented by the exception was moved from March 25, 2008 to March 26, 2008.

cb: 56

lpb:

040000008200E00074C5B7101A82E00807D803195025D461E473C801000000000000000
100000002A5844B3A444F74A9C246C60886F116B

Name	Type	Size	Sample	Description
Identifier	**Byte Array**	16	04 00 00 00 82 00 E0 00 74 C5 B7 10 1A 82 E0 08	This byte array identifies the BLOB as a Global Object ID.
Year	**WORD**	2	07 D8	The original year of the Instance represented by the exception. This value is in big-endian 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 uniquely identifying this meeting object.

4.1.2.2 PidLidCleanGlobalObjectId

The following is the value of the PidLidCleanGlobalObjectId **property** for the **exception** from example 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 zero.

cb: 56

lpb:

040000008200E00074C5B7101A82E008**00000000**5025D461E473C801000000000000000
100000002A5844B3A444F74A9C246C60886F116B

4.1.3 Sample 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 don't understand the format, as specified in section 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
```

The following shows how the body of the Meeting Request Object might look to a client that doesn't 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
```

4.1.4 Sample TimeZoneDefinition BLOB

The following is a sample Time Zone Definition BLOB.

```
cb: 184 (0x000000B8)
lpb:
0201300002001500500061006300690066006900630020005300740061006E006400
6100720064002000540069006D00650002002013E000000D607000000000000000
0000000000000E001000000000000C4FFFFFF00000A000000050002000000000000
000000040000000100020000000000000002013E000200D7070000000000000000000
0000000000E001000000000000C4FFFFFF00000B0000000100020000000000000000
00003000000002000200000000000000
```

The following table describes the content of this TimeZoneDefinition BLOB.

Name	Type	Size	Sample	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** characters.

Name	Type	Size	Sample	Description
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 Standard 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.
lDaylightBias	**LONG**	4	C4 FF FF FF	Daylight offset of -60 from the standard bias during daylight time.

Name	Type	Size	Sample	Description
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 meaning that the time zone will transition to standard time on the last Sunday of October at 2 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): wYear: 0 wMonth: 4 wDayOfWeek: 0 wDay: 1 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 meaning that the time zone will transition to daylight time on the first Sunday of April at 2 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 indicating that this rule is the effective rule.
wYear	**WORD**	2	D7 07	This rule is applicable beginning January 1, 2007.

Name	Type	Size	Sample	Description
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 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 meaning that the time zone will transition to standard time on the first Sunday of November at 2 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): wYear: 0 wMonth: 3 wDayOfWeek: 0 wDay: 2 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 meaning that the time zone will transition to daylight time on the second Sunday of March at 2 A.M.

4.1.5 Sample of PidLidTimeZoneStruct

The following is a sample value for the PidLidTimeZoneStruct.

```
cb: 48 (0x00000030)
lpb:
E001000000000000C4FFFFFF000000000B00000001000200000000000000000000
0030000000200020000000000000
```

The following table describes the content of the sample PidLidTimeZoneStruct BLOB.

Name	Type	Size	Sample	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 indicating 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): wYear: 0 wMonth: 11 wDayOfWeek: 0 wDay: 1 wHour: 2 wMinute: 0 wSecond: 0 wMilliseconds: 0 meaning that the time zone will transition to standard time on the first Sunday of November at 2am.
wDaylightYear	**WORD**	2	00 00	No year is specified indicating that the rule is a relative rule.

Name	Type	Size	Sample	Description
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 meaning that the time zone will transition to daylight time on the second Sunday of March at 2 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 A.M. It has a Daylight Date of {3, 0, 2, 2}, equivalent to the second Sunday of March at 2 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

Property	Property set GUID	Name or ID
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
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

Property	Property set GUID	Name or ID
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
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

Property	Property set GUID	Name or ID
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
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.

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

Property	Property ID
PidLidFInvited	0x81DA
PidLidAppointmentReplyName	0x81AE
PidLidRecurrenceType	0x81FE
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
PidLidTimeZone	0x8212
PidLidOwnerCriticalChange	0x8128
PidLidCalendarType	0x81B7
PidLidCleanGlobalObjectId	0x81B8
PidLidAppointmentMessageClass	0x8311

Property	Property ID
PidLidMeetingType	0x8314
PidLidOldLocation	0x8316
PidLidOldWhenEndWhole	0x83CD
PidLidOldWhenStartWhole	0x83CC

4.2.1.1 Sample Appointment

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 a half an hour before it happens. She wants to treat this as a private appointment, indicating 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 is 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 17:00:00.000)
PidLidCommonEnd	0x81bb	0x0040 (PtypTime)	0x01c9ca86a5089000 (2009/05/01 18:00:00.000)

Property	Property ID	Property type	Value
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 17:00:00.000)

Property	Property ID	Property type	Value
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 a sample explaining this TimeZoneDefinition BLOB. The time zone data for this Appointment is the following:

cb: 184

lpb:

```
0201300002001500500061006300690066006900630020005300740061006E006400
6100720064002000540069006D006500020002013E000000D6070000000000000000
000000000000E001000000000000C4FFFFFF00000A000000050002000000000000
000000040000000100020000000000000002013E000200D707000000000000000000
0000000000E001000000000000C4FFFFFF00000B000000010002000000000000000
0000300000002000200000000000000
```

*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 a sample explaining the Global Obj ID BLOB. The following is the value for this Appointment:

cb: 56

lpb:
040000008200E00074C5B7101A82E0080000000020631F30F072C8010000000000
00000010000000D97737CAB6762A43BFF793851D08DB16

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 indicating that the data has been accepted.

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

4.2.1.2 Sample Meeting

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 is a description of what a client might do to accomplish these tasks and the responses a server might return.

4.2.1.2.1 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 is 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)

Property	Property ID	Property type	Value
PidLidAppointmentStartWhole	0x81B2	0x0040 (PtypTime)	0x01C878A5984A4400 (2008/02/26 18:30:00.000)
PidLidAppointmentEndWhole	0x81AC	0x0040 (PtypTime)	0x01C878A9C92C7800 (2008/02/26 19:00:00.000)
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 AM to 11:00 AM
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)

Property	Property ID	Property type	Value
PidLidReminderDelta	0x81FF	0x0003 (PtypInteger32)	0x0000000F (15)
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, indicating to Mr. Saylor the purpose of the meeting. See [MS-OXBBODY] for details.		

*1 See section 4.1.5 for a sample explaining the PidLidTimeZoneStruct BLOB. The following is the value for this Meeting Object:

```
cb: 48

lpb:
E0010000000000C4FFFFFF000000000B0000000100020000000000000000000
0030000000200020000000000000
```

*2 The PidLidAppointmentTimeZoneDefinitionRecur dates for this **Appointment** are both set in the same time zone. See section 4.1.4 for a sample explaining 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:
0201300002001500500061006300690066006900630020005300740061006E006400
6100720064002000540069006D006500020002013E000000D607000000000000000
000000000000E0010000000000C4FFFFFF00000A000000050002000000000000
0000000400000001000200000000000000002013E000300D707000000000000000000
0000000000E0010000000000C4FFFFFF00000B00000001000200000000000000
0000300000002000200000000000000
```

*3 The start and end dates for this Appointment are both set in the same time zone. See section 4.1.4 for a sample explaining the TimeZoneDefinition BLOB. The following is the value for this Meeting Object:

```
cb: 184

lpb:
0201300002001500500061006300690066006900630020005300740061006E006400
61007200640020005400690006D006500020002013E000000D607000000000000000
000000000000E0010000000000C4FFFFFF00000A000000050002000000000000
0000000400000001000200000000000000002013E000200D707000000000000000000
0000000000E0010000000000C4FFFFFF00000B00000001000200000000000000
0000300000002000200000000000000
```

*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 sample explaining the Global Obj ID BLOB. The following is the value for this Meeting Object:

> cb: 56
>
> lpb:
> 040000008200E00074C5B7101A82E00800000000406FD661E473C8010000000000
> 000000100000002A5844B3A444F74A9C246C60886F116B

*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:
> 043004300B2001000000C0210000010000000000000004000000232000000A00000
> 0000000000000000000000000002088C30CDF80E95A063000000930000076020000 9
> 402000000000000000000000000

The client uses the **RopModifyRecipients** to add Dennis Saylor to the Meeting Object, including the following Extra Properties.

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 this, the newly created object will not be persisted. The server returns a success code indicating that the data has been accepted.

4.2.1.2.2 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, onto this new Meeting Request Object, all of the properties that were set on the **Meeting Object** in section 4.2.1.2.1 except for the following:

- PidLidBusyStatus

- PidLidAppointmentStateFlags
- PidLidResponseStatus
- PidLidFInvited
- PidLidAppointmentSequence
- PidLidAutoFillLocation
- PidLidReminderDelta*
- PidLidReminderSignalTime*
- PidLidSideEffects
- PidTagMessageClass
- PidTagIconIndex
- Best Body Properties

*The value 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 following **property** values 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

Property	Property ID	Property type	Value
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)
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 arrive to the Attendee. This client also needs to use **RopModifyRecipients** to add a RecipientRow for Mr. Saylor to the Meeting Request Object.

Once 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 following changes to the Meeting Object on Mr. John's calendar with **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 (PtypTime)	0x01C874276FF4F450 (2008/02/21 01:16:51.093)
PidTagOwnerAppointmentId	0x0062	0x0003 (PtypInteger32)	0x4D9427D8 (1301555160)

Finally, the client issues the **RopSaveChangesMessage** to save these changes to the Organizer's Meeting Object, and then releases both the Meeting and **Meeting Request Objects** with a **RopRelease** for each.

4.2.1.2.3 Receiving the Meeting Request

Upon receiving the **Meeting Request Object** that was sent in the example of section 4.2.1.2.2, a client might tentatively add a **Meeting Object** to the **Calendar Special Folder** in Mr. Saylor's mailbox. The following describes what a client might do to accomplish this task.

The client uses **RopOpenMessage** 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, onto this new Meeting Object, all of the properties that were set on the Meeting Request Object in section 4.2.1.2.2 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 following **property** values onto the Meeting Object.

Property	Property ID	Property type	Value

Property	Property ID	Property type	Value
PidLidReminderDelta	0x81FF	0x0003 (PtypInteger32)	0x0000000F (15) The default value for this client, since 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)
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 with **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 indicating that the data has been accepted.

The client sets the following properties on the Meeting Request Object 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.

4.2.1.2.4 Accepting the Meeting Request

Upon receiving the **Meeting Request Object** that was sent in the example of section 4.2.1.2.2, 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 Mr. Saylor will be in attendance. The following describes what a client might do to accomplish this task.

The client uses **RopOpenMessage** to obtain a handle to the tentative **Meeting Object** that had been created in section 4.2.1.2.3, 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 following properties are then modified on the new Meeting Object using **RopSetProperties**:

Property	**Property ID**	**Property type**	**Value**
PidLidAppointmentMessageClass	0x8311	0x001F (PtypString)	IPM.Appointment
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000002 (olBusy)
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000003 (respAccepted)
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 handle to the tentative Meeting Object with **RopRelease** and then deletes the tentative Meeting Object with **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 **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 following property values 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 using **RopModifyRecipients** and then sends the object via **RopSubmit**. After the server returns a success code from submission, the client releases both the Meeting and **Meeting Response Objects** with a **RopRelease** for each.

4.2.1.2.5 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**. The following describes what a client might do 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".

Upon 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 among the RecipientRows, first attempting to find a PidTagSearchKey that matches the PidTagSentRepresentingSearchKey (or PidTagSenderSearchKey). If no match was found, then the client attempts to match the PidTagDisplayName property from the RecipientRow with PidTagSentRepresentingName (or PidTagSenderName).

If a RecipientRow was not found, then a new one with Recipient Type RECIP_CC is added to the in-memory recipient cache to represent this **Attendee**. The following Extra Properties are added to the in-memory RecipientRow representing this attendee:

Property	**Property ID**	**Property type**	**Value**
PidTagRecipientTrackStatus	0x5FFF	0x0003 (PtypInteger32)	0x00000003 (respAccepted)

Property	Property ID	Property type	Value
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 **RopModifyRecipients** to copy the in-memory recipient cache back onto the message object.

The client sets the following properties on the Meeting Request Object 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 with **RopRelease**.

4.2.1.2.6 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 sends a **Meeting Update Object** to notify Mr. Saylor of the new date. The following is a description of what a client might do to accomplish this task and the responses a server might return.

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 following **property** is set 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 following properties 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)
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)

Property	Property ID	Property type	Value
PidTagStartDate	0x0060	0x0040 (PtypTime)	0x01C88E9DDA16DC00 (2008/03/25 17:30:00.000)
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 sample explaining the TimeZoneDefinition BLOB. The following is the value for this exception (and is the same as the associated Meeting Object).

cb: 184

lpb:
```
0201300002001500500061006300690066006900630020005300740061006E006400
6100720064002000540069006D006500020002013E000000D6070000000000000000
0000000000000E00100000000000C4FFFFFF00000A00000005000200000000000000
00000004000000010002000000000000000002013E000200D707000000000000000000
0000000000E00100000000000C4FFFFFF00000B00000001000200000000000000000
000030000000200020000000000000
```

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 with **RopSaveChangesMessage**, to which the server returns success codes.

The client uses **RopSetProperties** to set the following 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.

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, onto 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)
PidLidResponseStatus	0x8122	0x0003 (PtypInteger32)	0x00000005 (respNotResponded)

Property	Property ID	Property type	Value
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 AM to 11:00 AM
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 (PtypTime)	0x01C88F6704809C00 (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

Property	Property ID	Property type	Value
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)
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 sample explaining the PidLidTimeZoneStruct BLOB. The following is the value for this Meeting Request Object:

cb: 48

lpb:
E001000000000000C4FFFFFF000000000B0000000100020000000000000000000000300000000200020000000000000

*2 The PidLidAppointmentTimeZoneDefinitionRecur dates for this Appointment are both set in the same time zone. See section 4.1.4 for a sample explaining 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:
```
0201300002001500500061006300690066006900630020005300740061006E006400
610072006400200054006900 6D006500020002013E000000D607000000000000000
```

```
lpb:
040000008200E00074C5B7101A82E008000000005025D461E473C8010000000000
000000100000002A5844B3A444F74A9C246C60886F116B
```

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 arrive 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 following changes to the Meeting Object (the object representing the **Recurring Series**) on Mr. John's calendar with **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:
043004300B2001000000C021000001000000000000000040000000232000000A00000
000000000001000000A025C40C01000000402BC40C2088C30CDF80E95A0630000
00930000007602000094020000100B62DC40CD42DC40C1628C40C000200000000
040000000000000000000000000000
```

Finally, the client issues the **RopSaveChangesMessage** to save the Meeting Object representing the Recurring Series, and then uses **RopRelease** to release all handles (embedded message, attachment, meeting, **Meeting Request Objects**).

4.2.1.2.7 Accepting the Exception

Upon receiving the **Meeting Update Object** that was sent in the example of section 4.2.1.2.6, Mr. Dennis Saylor decides 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. The following describes what a client might do to accomplish this task and the responses that a server might return.

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, 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 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 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 in the row, until a match is found.<104> Upon finding a matching row, the client issues RopOpenMessage 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.<105>

For this example, an Exception Attachment Object did 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 on the **attachment object**.

Property	Property ID	Property type	Value

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 following properties 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
PidLidAppointmentDuration	0x81A9	0x0003 (PtypInteger32)	0x0000001E (30)

Property	Property ID	Property type	Value
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 AM to 11:00 AM
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)
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)

Property	Property ID	Property type	Value
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 (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)
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 sample explaining the PidLidTimeZoneStruct BLOB. The following is the value for this Meeting Request Object:

cb: 48

lpb:
E001000000000000C4FFFFFF000000000B0000000100020000000000000000000
00300000002000200000000000000

*2 The PidLidAppointmentTimeZoneDefinitionRecur dates for this **Appointment** are both set in the same time zone. See section 4.1.4 for a sample explaining 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:
0201300002001500500061006300690066006900630020005300740061006E006400
610072006400200054006900 6D00650002000201 3E000000D607000000000000000
```
0201300002001500500061006300690066006900630020005300740061006E006400
6100720064002000540069006D006500020002013E000000D607000000000000000
0000000000000E001000000000000C4FFFFFF00000A0000000500020000000000000
0000000400000001000200000000000000002013E000300D70700000000000000000
0000000000E001000000000000C4FFFFFF00000B00000001000200000000000000000
00003000000020002000000000000000
```

*3 The start and end dates for this Appointment are both set in the same time zone. See section 4.1.4 for a sample explaining the TimeZoneDefinition BLOB. The following is the value for this Meeting Request Object:

cb: 184

lpb:
```
0201300002001500500061006300690066006900630020005300740061006E006400
6100720064002000540069006D006500020002013E000000D607000000000000000
0000000000000E001000000000000C4FFFFFF00000A0000000500020000000000000
0000000400000001000200000000000000002013E000200D70700000000000000000
0000000000E001000000000000C4FFFFFF00000B00000001000200000000000000000
00003000000020002000000000000000
```

*4 The following is the value of the PidLidGlobalObjectId for this Meeting Request Object. See section 4.1.2 for a sample explaining the Global Obj ID BLOB.

cb: 56

lpb:
```
040000008200E00074C5B7101A82E00807D803195025D461E473C801000000000
0000000100000002A5844B3A444F74A9C246C60886F116B
```

*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:
040000008200E00074C5B7101A82E008000000005025D461E473C8010000000000000000
100000002A5844B3A444F74A9C246C60886F116B

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 using **RopSaveChangesMessage**, to which the server returns a success code.

After saving the Exception Embedded Message Object, the client uses **RopSetProperties** to set the following 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 changes to the Meeting Object (the object representing the Recurring Series) on Mr. Saylor's calendar with **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:
043004300B2001000000C0210000010000000000000004000000232000000A00000
00000000001000000A025C40C01000000402BC40C2088C30CDF80E95A0630000
0093000007602000094020000000100B62DC40CD42DC40C1628C40C000200000000
04000000000000000000000000000000
```

The client sets the following properties on the Meeting Request Object using **RopSetProperties**, followed by **RopSaveChangesMessage**.

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

Having processed the Meeting Request Object, the client is now ready to act on the response. To start, the following changes are made to the Exception Embedded Message Object using RopSetProperties.

Property	Property ID	Property type	Value
PidLidBusyStatus	0x81B6	0x0003 (PtypInteger32)	0x00000002 (2)
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 with **RopSaveChangesMessage** and another **RopSaveChangesMessage** to save the Meeting Object representing 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** with RopCreateMessage, to which the server returns a success code and a handle. The client sets the following properties on this new message object with 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 following property values 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:

Property	Property ID	Property type	Value
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 using **RopModifyRecipients** and then sends the object via **RopSubmit**. 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 with 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 pertaining to the underlying RPC-based transport apply (see [MS-OXCROPS]).

5.2 Index of Security Parameters

None.

6 Appendix A: Office/Exchange Behavior

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

- Office 2003 with Service Pack 3 applied
- Exchange 2003 with Service Pack 2 applied
- Office 2007 with Service Pack 1 applied
- Exchange 2007 with Service Pack 1 applied

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

<1> Office 2003 SP3 and Office 2007 SP1 sets the following additional properties on a new object regardless of user input.

PidLidAgingDontAgeMe, PidLidCurrentVersion, PidLidCurrentVersionName, PidLidValidFlagStringProof, PidTagAlternateRecipientAllowed, PidTagClientSubmitTime, PidTagDeleteAfterSubmit,, PidTagMessageDeliveryTime, PidTagOriginatorDeliveryReportRequested, PidTagReadReceiptRequested
<2> The following additional properties can be set on items described by the Appointment and Meeting Object protocol for backward compatibility with older clients. These properties are not used by Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, or Exchange 2007 SP1.
PidLidRequiredAttendees, PidLidOptionalAttendees, PidLidResourceAttendees, PidLidDelegateMail, PidLidSingleInvite, PidLidTimeZone, PidLidStartRecurDate, PidLidStartRecurTime, PidLidEndRecurDate, PidLidEndRecurTime, PidLidDayInterval, PidLidWeekInterval, PidLidMonthInterval, PidLidYearInterval, PidLidDowMask, PidLidDomMask, PidLidMoyMask, PidLidRecurrenceType, PidLidDowPref, PidLidAllAttendeesList.
<3> Office 2007 SP1 sets the following properties regardless of user input; their values have no meaning in the context of this protocol.
PidLidTaskStatus, PidLidPercentComplete, PidLidTaskSMUG, PidLidTaskActualEffort, PidLidTaskEstimatedEffort, PidLidTaskVersion, PidLidTaskState, PidLidTaskComplete, PidLidTaskAssigner, PidLidTaskOrdinal, PidLidTaskNoCompute, PidLidTaskFRecur, PidLidTaskRole, PidLidTaskOwnership, PidLidTaskAcceptanceState, PidLidTaskFFixOffline.
<4> Exchange 2003 SP3 and Exchange 2007 SP1 do not set the auxApptFlagCopied flag when copying calendar objects.
<5> Exchange 2003 SP2 and Exchange 2007 SP1 do not respect the auxApptFlagForceMtgResponse bit of the PidLidAppointmentAuxFlags property.
Office 2007 SP1 respects this bit when the following Registry Value is set to a nonzero value:
Key: HKCU\Software\Microsoft\Office\12.0\Outlook\Options\Calendar
DWORD Value: ForceMtgForwardResponse
Office 2003 SP3 respects this bit when the following Registry Value is set to a nonzero value:
Key: HKCU\Software\Microsoft\Office\11.0\Outlook\Options\Calendar
DWORD Value: ForceMtgForwardResponse
<6> Exchange 2003 SP3 ignores this property and always computes this from the difference between PidLidAppointmentEndWhole and PidLidAppointmentStartWhole.
<7> Exchange 2003 SP3 does not read or write this property.
<8> Office 2007 SP1 and Exchange 2007 SP1 use the PidLidAppointmentUnsendableRecipients property if it exists, and will only use PidLidNonSendableTo in the absence of that one.
<9 > Office 2007 SP1 and Exchange 2007 SP1 use the PidLidAppointmentUnsendableRecipients property if it exists, and will only use PidLidNonSendableCc in the absence of that one.
<10> Office 2007 SP1 and Exchange 2007 SP1 use the PidLidAppointmentUnsendableRecipients property if it exists, and will only use PidLidNonSendableBcc in the absence of that one.

<11> Office 2007 SP1 and Exchange 2007 SP1 use the PidLidAppointmentUnsendableRecipients property if it exists, and will only use PidLidNonSendableToTrackStatus in the absence of that one.
<12> Office 2007 SP1 and Exchange 2007 SP1 use the PidLidAppointmentUnsendableRecipients property if it exists, and will only use PidLidNonSendableCcTrackStatus in the absence of that one.
<13> Office 2007 SP1 and Exchange 2007 SP1 use the PidLidAppointmentUnsendableRecipients property if it exists, and will only use PidLidNonSendableBccTrackStatus in the absence of that one.
<14> Office 2007 SP1 and Exchange 2007 SP1 use PidLidAppointmentUnsendableRecipients to keep track of Unsendable Attendees. Office 2003 SP3 and Exchange 2003 SP2 do not, but instead use the following properties (these are written by Office 2007 SP1 and Exchange 2007 SP1 for backward compatibility only):
PidLidNonSendableTo
PidLidNonSendableCc
PidLidNonSendableBcc
PidLidNonSendableToTrackStatus
PidLidNonSendableCcTrackStatus
PidLidNonSendableBccTrackStatus
<15> When a meeting object is created, Office 2003 SP3 and Office 2007 SP1 set this value to the number of minutes between the start time and midnight, January 1, 1601. When trying to find a meeting object, Office 2003 SP3 and Office 2007 SP1 sort the table according to the PidLidOwnerAppointmentId property, thus allowing increased performance in the search.
<16> Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, and Exchange 2007 SP1 allow the user to choose whether or not they want to send a Meeting Response Object to the organizer.
<17> 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, then the properties PidLidAppointmentTimeZoneDefinitionRecur and PidLidTimeZoneStruct are considered inconsistent.
<18> Office 2003 SP3 does not support PidLidAppointmentTimeZoneDefinitionRecur.
<19> 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

<20> Office 2003 SP3 does not support PidLidAppointmentTimeZoneDefinitionStartDisplay.
<21> Office 2003 SP3 does not support PidLidAppointmentTimeZoneDefinitionEndDisplay.
<22> Exchange 2003 SP2 and Exchange 2007 SP1 use the signal time rather than the start time when calculating whether or not exceptions overlap. Office 2003 SP3 and Office 2007 SP1 use the start time.
<23> Exchange 2003 SP2 supports only the Gregorian calendar. Exchange 2007 SP1 does not support the CAL_SAKA calendar.

<24> 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). 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, as (Day X modulo 1440) will always yield 0, indicating 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, indicating 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 = 11520
Check 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 and (period - 1440).
<25> 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 the pattern was created with. 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 table might make this a little bit easier to see:

Assuming a pattern "Every 2 weeks on Mon, Tue, and Fri., Starting in week 2"

Week	First Day of Week is Sunday							First Day of Week is Wednesday						
	Su	Mo	Tu	We	Th	Fr	Sa	We	Th	Fr	Sa	Su	Mo	Tu
1	1	2	3	4	5	6	7	4	5	6	7	8	9	10
2	8	**9**	**10**	11	12	**13**	14	11	12	**13**	14	15	**16**	**17**
3	15	16	17	18	19	20	21	18	19	20	21	22	23	24
4	22	**23**	**24**	25	26	**27**	28	25	26	**27**	28	29	**30**	**31**

As can be seen, if the first day of the week were 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 were 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. Once 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 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 instance:

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 example above (week starts on Wednesday), assume that the input date provided was the 21st.

1. Adjust to start of week, which is the 18th.
2. Using the FirstDateTime weekly offset value, determine if this is a valid week. If 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.

<26> The following is a description of how the FirstDateTime value is used for a Montly or Yearly recurrence pattern.

Monthly and Yearly are evaluated in the same manner; 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 straight forward. 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.

There are some other details to deal with non-Gregorian calendars, which may have leap months and other non-Gregorian specific details.

<27> Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, and Exchange 2007 SP1 always write a default value of 0x0000000A for the Occurrence Count when the recurrence pattern has no end date.

<28> Exchange 2007 SP1 does not allow duplicate entries, and will remove them if they are present.

<29> Exchange 2007 SP1 does not allow duplicate entries, and will remove them if they are present.

<30> This flag is not set in Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, or Exchange 2007 SP1. This flag is reserved for future enhancements and MUST NOT be used.

<31> This field does not exist in Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, or Exchange 2007 SP1. This field is reserved for future enhancements and MUST NOT be used.

<32> Office 2007 SP1 sets this property but Office 2003 SP3, Exchange 2003 SP2, and Exchange 2007 SP1 do not.

<33> Exchange 2003 SP2 does not read or write this property, but Office 2003 SP3, Office 2007 SP1, and Exchange 2007 SP2 do.

<34> Office 2003 SP3 reads and writes the properties in this section. Office 2007 SP1 does not write any of these properties but reads some of them. Exchange 2003 SP2 and Exchange 2007 SP1 do not read or write these properties.

<35> Calendar objects can also have the following reminder-related properties as specified in the [MS-OXORMDR] protocol:

PidLidReminderSet, PidLidReminderSignalTime, PidLidReminderDelta, PidLidReminderTime, PidLidReminderOverride, PidLidReminderPlaySound, PidLidReminderFileParam.

<36> Exchange 2003 SP2 only includes the seCoerceToInbox and seOpenForCtxMenu flags. Without all the flags, the Outlook UI 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.

<37> The PidLidFExceptionalAttendees property is utilized to determine, from an Appointment object, if Attendees have been invited to any exceptions.
<38> Meeting Objects can also have the following properties:
PidLidOrigStoreEidCalendar.
<39> If there is more than one resource in a meeting object, the PidLidLocation property is set to the first sendable resource added to the meeting. If none of the resources are sendable, the PidLidLocation property is set to the first unsendable resource added to the meeting.
<40> Office 2003 SP3 and Office 2007 SP1 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.
<41> Office 2003 SP3 and Office 2007 SP1 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.
<42> Office 2003 SP3 and Office 2007 SP1 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.
<43> If this value is not specified, Exchange 2003 SP2 will assume the last modified time as this value. Exchange 2007 SP1, Office 2003 SP3, and Office 2007 SP1 do not make this assumption.
<44> Exchange 2003 SP2 does not read or write this property.
<45> The data in this table is used by Office 2003 SP3 and Office 2007 SP1, although its content is subject to change with future time zone updates.
<46> Meeting Request and Update objects can also have the following properties which have no effect on the Appointment and Meeting Object protocol:
PidLidTrustRecipHighlights.
<47> Exchange 2003 SP2 and Outlook 2003 SP3 do not read or write this property.
<48> The property PidLidForwardInstance is used by Office 2003 SP3, but not by Office 2007 SP1, Exchange 2003 SP2, or Exchange 2007 SP1.
<49> Office 2007 SP1 and Exchange 2007 SP1 set this property but Office 2003 SP3 and Exchange 2003 SP2 do not.
<50> Office 2007 SP1 and Exchange 2007 SP1 set this property but Office 2003 SP3 and Exchange 2003 SP2 do not.
<51> Office 2003 SP3 and Exchange 2003 SP2 set this property but Office 2003 SP3 and Exchange 2003 SP2 do not.
<52> Office 2003 SP3 and Office 2007 SP1 show the values of the PidLidAppointmentStartWhole, PidLidAppointmentEndWhole, and PidLidLocation properties as the Downlevel Text. Exchange 2003 SP3 and Exchange 2007 SP1 do not add Downlevel Text.
<53> For English, Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, and Exchange 2007 SP1 use the string “New Time Proposed:” to indicate that the Meeting Response Object includes a new date/time proposal. If no proposal is included, Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, and Exchange 2007 SP1 use “Accepted:”, “Tentative:”, or “Declined:” for an accepted, tentatively accepted, or declined meeting response, respectively.

<54> For English, Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, and Exchange 2007 SP1 use the string “Canceled:”.
<55> 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. The following are two cases that sometimes occur with Office:

- When an attendee forwards a meeting request object to another attendee, the new attendee receives the information for the recurring series (including the PidLidAppointmentRecur property) but does not receive the exception attachment objects. In this case, there are fewer exception attachment objects (for example, none) than values in the ModifiedInstanceDates field.
- 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.

<56> If the user changes the client machine’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.
<57> If the user changes the client machine’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.
<58> Office 2003 SP3 and Office 2007 SP1 does not write this value.
<59> An end user can create calendar items in any Calendar Folder. However, free/busy information is only calculated from the Calendar Special Folder.
<60> When an end user creates a meeting in a Calendar Folder other than the Calendar Special Folder, Office will ask the user if he or she wants to create a clone in the Calendar Special Folder. Exchange will not create a clone of the meeting.
<61> 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.
<62> Office sometimes does not copy the recipient list. If the RecipientRows from a meeting object are not copied, then the resulting snapshot will not show who was invited to the meeting at the time the copy was made.
<63> Office 2007 SP1 and Exchange 2007 SP1 require the organizer to send a meeting cancelation to Attendees when deleting a meeting. Office 2003 SP3 and Exchange 2003 SP2 give the user an option to delete without sending a cancelation.
<64> Office attempts Direct Booking only for resources. Exchange does not attempt Direct Booking for any Attendees.
<65> This requires public folders to be enabled on the server. Exchange 2007 SP1 allows a configuration without public folders, in which case direct booking would not be possible.
<66> Office 2007 SP1 and Exchange 2007 SP1 support the Calendar Dictionary, but Office 2003 SP3 and Exchange 2003 SP2 do not.
<67> A private meeting request object will have the value of the PidTagSensitivity property (see [MS-OXCMSG]) set to private.

<68> Office 2007 SP1 respects the PidTagScheduleInfoDelegatorWantsInfo property, but Office 2003 SP3, Exchange 2003 SP2, and Exchange 2007 SP1 do not.
<69> Office 2003 SP3 and Office 2007 SP1 do this in certain circumstances. Exchange 2003 SP2 and Exchange 2007 SP1 never change the PidTagMessageClass property in this manner.
<70> Office 2003 SP3 and Office 2007 SP1 both copy the PidLidAppointmentAuxFlags to the meeting object but Exchange 2003 SP2 and Exchange 2007 SP1 do not.
<71> Office 2003 SP3 and Office 2007 SP1 both set PidTagProcessed. Exchange 2003 SP2 and Exchange 2007 SP1 do not set this flag.
<72> Office 2007 SP1 and Exchange 2007 SP1 will set the "old" properties. Office 2003 SP3 and Exchange 2003 SP2 will not set these.
<73> Office 2007 SP1 and Exchange 2007 SP1 will set the value of the PidLidMeetingType to mtgInfo in this case. Office 2003 and Exchange 2003 SP2 will set the value of this property to mtgFull.
<74> Office 2003 SP3 and Exchange 2003 SP2 will always clear responses whenever any update is sent out.
<75> Office 2003 SP3 and Office 2007 SP1 set the PidTagRecipientTrackStatusTime value to 12:18 a.m. 23 October 1602. Exchange 2003 SP2 and Exchange 2007 SP1 do not change this value. Changing this value is not required.
<76> Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, and Exchange 2007 SP1 both give the user a choice on whether they want to send the update to all recipients or only added/removed recipients.
<77> Office 2007 SP1 and Exchange 2007 SP1 set the PidLidAppointmentUnsendableRecipients as described, while Office 2003 SP3 and Exchange 2003 SP2 do not.
<78> Office 2007 and Exchange 2007 SP1 support the Calendar Dictionary, but Office 2003 SP3 and Exchange 2003 SP2 do not.
<79> A private meeting request object will have the value of the PidTagSensitivity property (see [MS-OXCMSG]) set to 0x00000002.
<80> Office 2007 SP1 respects the PidTagScheduleInfoDelegatorWantsInfo property, but Office 2003 SP3, Exchange 2003 SP2, and Exchange 2007 SP1 do not.
<81> Office 2007 SP1 copies these properties onto the meeting update object, while Office 2003 SP3, Exchange 2003 SP2, and Exchange 2007 SP1 do not.
<82> Office 2007 SP1 and Exchange 2007 SP1 allow a meeting object to be updated without changing the value of the PidLidResponseStatus property. Office 2003 SP3 and Exchange 2003 SP2 reset the value of this property to respNotResponded.
<83> Office 2003 SP3 and Office 2007 SP1 both set PidTagProcessed. Exchange 2003 SP2, and Exchange 2007 SP1 do not set this flag.
<84> Office 2007 SP1 and Exchange 2007 SP1 write the PidLidAppointmentUnsendableRecipients property, but Office 2003SP3 and Exchange 2003 SP2 do not.
<85> Exchange 2003 SP2 and Exchange 2007 SP1 never set the auxApptFlagForceMtgResponse bit in the PidLidAppointmentAuxFlags property.

Office 2007 SP1 will set this bit on a forwarded meeting request when the following Registry Value is set to a nonzero value:
Key: HKCU\Software\Microsoft\Office\12.0\Outlook\Options\Calendar
DWORD Value: ForceMtgForwardResponse
Office 2003 SP3 will set this bit on a forwarded meeting request when the following Registry Value is set to a nonzero value:
Key: HKCU\Software\Microsoft\Office\11.0\Outlook\Options\Calendar
DWORD Value: ForceMtgForwardResponse
<86> When a meeting request object is forwarded to another user, and the object is sent through an Exchange 2007 SP1 server, Exchange 2007 SP1 creates what is called a Meeting Forward Notification object (MFN) and sends it to the organizer, notifying him or her of the new Attendees. Exchange 2007 SP1 recognizes the meeting request object as a forwarded object by the presence of the auxApptFlagForwarded flag in the value of the PidLidAppointmentAuxFlags property. When Exchange 2007 SP1 receives an MFN, it adds the Attendees as RecipientRows in the organizer's meeting object, and then moves the MFN to the deleted items special folder.
<87> Office 2003 SP3 and Office 2007 SP1 creates a copy and modifies the copy, unless a certain registry key is set. Exchange 2003 SP2, Exchange 2007 SP1 always creates and modifies a copy.
<88> Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, and Exchange 2007 SP1 allow the end user to decide whether or not the end user wants to send a response to the organizer.
<89> Office 2003 SP3 and Exchange 2003 SP2 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. Office 2007 SP1 and Exchange 2007 SP1 will not allow an organizer to respond to their own meeting.
<90> Often when the organizer sends a meeting request object to a very large set of people, the organizer does not wish 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.
<91> Exchange 2003 SP2 and Exchange 2007 SP1 do not pay attention to the PidLidAppointmentAuxFlags property.
Office 2007 SP1 will force a meeting request object to be sent to the user when the auxApptFlagForceMtgResponse bit is set, and when the following Registry Value is set to a nonzero value:
Key: HKCU\Software\Microsoft\Office\12.0\Outlook\Options\Calendar
DWORD Value: ForceMtgForwardResponse
Office 2003 SP3 will force a meeting request object to be sent to the user when the auxApptFlagForceMtgResponse bit is set, and when the following Registry Value is set to a nonzero value:
Key: HKCU\Software\Microsoft\Office\11.0\Outlook\Options\Calendar
DWORD Value: ForceMtgForwardResponse
<92> Office 2003 SP3 and Office 2007 SP1 also writes the following properties which are not used by Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, or Exchange 2007 SP1:

PidLidInetAcctName, PidLidInetAcctStamp, PidLidSendMtgAsICAL
<93> Office 2003 SP3 and Office 2007 SP1 also writes the following properties when the Meeting Response Object represents a recurring series. These are not used by Office 2003 SP3, Office 2007 SP1, Exchange 2003 SP2, or Exchange 2007 SP1: PidLidRequiredAttendees, PidLidOptionalAttendees, PidLidResourceAttendees, PidLidDelegateMail, PidLidSingleInvite, PidLidTimeZone, PidLidStartRecurDate, PidLidStartRecurTime, PidLidEndRecurDate, PidLidEndRecurTime, PidLidDayInterval, PidLidWeekInterval, PidLidMonthInterval, PidLidYearInterval, PidLidDowMask, PidLidDomMask, PidLidMoyMask, PidLidRecurrenceType, PidLidDowPref, PidLidAllAttendeesList
<94> Office 2007 SP1 and Exchange 2007 SP1 support the Calendar Dictionary, but Office 2003 SP3 and Exchange 2003 SP2 do not.
<95> Office 2007 SP1 will recreate the exception to record the response. This causes the organizer to unexpectedly see the exception back in his or her calendar, often leading to confusion on the part of the organizer.
<96> Office 2003 SP3 and Office 2007 SP1 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 SP2 and Exchange 2007 SP1 do not round the time value.
<97> Office 2003 SP3 and Office 2007 SP1 round the time value from the PidLidAttendeeCriticalChange property down to the nearest minute before setting the value in the PidTagRecipientTrackStatusTime property. Exchange 2003 SP2 and Exchange 2007 SP1 do not round the time value.
<98> Office 2003 SP3 and Office 2007 SP1 allows the user to decide whether or not to "Delete empty responses". Exchange 2003 SP2 and Exchange 2007 SP1 never automatically deletes responses.
<99> Office 2007 SP1 and Exchange 2007 SP1 support the Calendar Dictionary, but Office 2003 SP3 and Exchange 2003 SP2 do not.
<100> Office 2003 SP3 and Office 2007 SP1 will recreate the Exception object, but Exchange 2003 SP2 and Exchange 2007 SP1 will not recreate the Exception object.
<101> Office 2003 SP3 and Office 2007 SP1 will create the meeting object but Exchange 2003 SP2 and Exchange 2007 SP1 will not create it.
<102> Office 2003 SP3 and Office 2007 SP1 both set PidTagProcessed. Exchange 2003 SP2 and Exchange 2007 SP1 do not set this flag.
<103> 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, then any Meeting Responses from the original Attendees will not be recorded on the meeting object. Exchange 2007 SP1 does set the new sequence number in the PidLidAppointmentSequence property.
<104> 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 (since this meeting update object represents an exception). If an Orphan Instance wasn't found, a client would search for a matching row with the PidTagOwnerAppointmentId value

of 0. If a matching recurring series or orphan exception still couldn't 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.

<105> 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, then 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.

Index