[MS-OXOTASK]:
Task-Related Objects Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Revised and edited technical content.
12/03/2008	1.03		Updated IP notice.
04/10/2009	2.0		Updated technical content and applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	5.0.0	Major	Updated and revised the technical content.
05/05/2010	5.0.1	Editorial	Revised and edited the technical content.
08/04/2010	6.0	Major	Significantly changed the technical content.
11/03/2010	6.1	Minor	Clarified the meaning of the technical content.
03/18/2011	6.1	No change	No changes to the meaning, language, and formatting of the technical content.
08/05/2011	7.0	Major	Significantly changed the technical content.
10/07/2011	7.1	Minor	Clarified the meaning of the technical content.
01/20/2012	8.0	Major	Significantly changed the technical content.
04/27/2012	8.1	Minor	Clarified the meaning of the technical content.
07/16/2012	8.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	8.2	Minor	Clarified the meaning of the technical content.
02/11/2013	8.2	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Task-Related Objects Protocol enables a user to assign a task to another user and track the progress. This protocol extends the Message and Attachment Object Protocol, which is described in [MS-OXCMSG].

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

1.1 Glossary

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

Coordinated Universal Time (UTC)
GUID
handle
Unicode

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

Attachment object
blind carbon copy (Bcc) recipient
carbon copy (Cc) recipient
contents table
delegate
display name
Embedded Message object
Folder object
long ID (LID)
mailbox
Message object
named property
primary recipient
property ID
property name
public folder
recurrence pattern
recurring task
remote operation (ROP)
restriction
Rich Text Format (RTF)
ROP request
store
tagged property
Task object
task request
Tasks folder

The following terms are specific to this document:

task acceptance: A Message object that is used to convey acceptance of a task assignment.

task assignee: A user to whom a task has been assigned.

task assigner: A user who assigns a task to another user.

task communication: Collectively, a task request, a task acceptance or task rejection, and a task update.

task owner: The user who is responsible for updating a task. For unassigned tasks, the local user is the owner. For assigned tasks, the task assignee is the owner.

task rejection: A Message object that is used to convey the rejection of a task assignment.

task update: A Message object that is used by a task assignee to send task changes to a task assigner.

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

1.2 References

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

1.2.1 Normative References

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

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

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

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

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

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

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

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

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

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

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

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

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

1.2.2 Informative References

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

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

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

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

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

1.3 Overview

The Task-Related Objects Protocol defines a **Task object** to represent a task that the user wants to accomplish. The properties of a Task object specify the due date, assignment status, and anticipated work effort, among other things. A Task object is created in the **Tasks folder** of a messaging **store**. Once a Task object is created, the task can be assigned. Task assignments are made, confirmed, and updated through the use of **task communications**, which include **task requests**, **task acceptances**, **task rejections**, and **task updates**. The Task-Related Objects Protocol also allows a series of tasks to be generated from a single Task object with a **recurrence pattern**.

This protocol extends the Message and Attachment Object Protocol, as described in [MS-OXCMSG], in that it defines new properties on a **Message object** and adds constraints to the existing properties of a Message object.

1.4 Relationship to Other Protocols

The Task-Related Objects Protocol extends the Message and Attachment Object Protocol, as described in [MS-OXCMSG], and, therefore, has the same dependencies.

The Task-Related Objects Protocol also depends on the Email Object Protocol, as described in [MS-OXOMSG], for sending a task communication.

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

1.5 Prerequisites/Preconditions

The Task-Related Objects Protocol has the same prerequisites and preconditions as the Message and Attachment Object Protocol, as described in [MS-OXCMSG].

1.6 Applicability Statement

A client can use this protocol to manage a user's tasks in the user's **mailbox**.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

This protocol provides no vendor extensibility beyond what is specified in [MS-OXCMSG].

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Task-Related Objects Protocol uses the same underlying transport as that used by the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].

2.2 Message Syntax

A Task object and a task communication can be created and modified by clients and servers. Except where noted, this section defines constraints under which both clients and servers operate.

Clients operate on a Task object and a task communication by using the Message and Attachment Object Protocol, as specified in [MS-OXCMSG]. How a server operates on a Task object and a task communication is implementation-dependent, but the results of any such operations MUST be exposed to clients in a manner that is consistent with the Task-Related Objects Protocol.

Unless otherwise specified in sections 2.2.1 through 2.2.3, a Task object and a task communication MUST adhere to all property constraints specified in [MS-OXPROPS] and [MS-OXCMSG].

2.2.1 Folder Properties

The property in section 2.2.1.1 is set on the **Folder object** that represents the Tasks folder. Task objects are stored in the Tasks folder.

2.2.1.1 PidTagOrdinalMost Property

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

The **PidTagOrdinalMost** property ([MS-OXPROPS] section 2.891) contains a positive number whose negative is less than or equal to the value of the **PidLidTaskOrdinal** property (section 2.2.2.2.26) of all Task objects in the folder. This property MUST be updated to maintain this condition whenever the **PidLidTaskOrdinal** property of any Task object in the folder changes in a way that would violate the condition.

This property provides an efficient way to identify Task objects as being in the same folder.

2.2.2 Task Object Properties

2.2.2.1 Additional Property Constraints

In some cases, the Task object has specific requirements for properties that are otherwise inherited. These specific requirements are specified in sections 2.2.2.1.1 through 2.2.2.1.5.

2.2.2.1.1 PidTagMessageClass Property

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

The **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) specifies the type of the Message object. The value MUST be "IPM.Task" or begin with "IPM.Task.". The string is case-insensitive.

2.2.2.1.2 Body Properties

The properties specified in [MS-OXCMSG] section 2.2.1.50 are used to specify **Rich Text Format (RTF)** for the body of a Task object.

2.2.2.1.3 PidLidCommonStart Property

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

The **PidLidCommonStart** property ([MS-OXCMSG] section 2.2.1.18) specifies the **Coordinated Universal Time (UTC)** equivalent of the **PidLidTaskStartDate** property (section 2.2.2.2.4).

2.2.2.1.4 PidLidCommonEnd Property

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

The **PidLidCommonEnd** property ([MS-OXCMSG] section 2.2.1.19) specifies the **UTC** equivalent of the **PidLidTaskDueDate** property (section 2.2.2.2.5).

2.2.2.1.5 PidTagIconIndex Property

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

The **PidTagIconIndex** property ([MS-OXOMSG] section 2.2.1.10) specifies which icon is to be used by a user interface to represent the Task object. If this property exists, its value is a hint to the client's user interface. The user interface can ignore the value and use another method of determining which icon to display to the user.

The value is one of the following.

Value	Meaning
0x00000501	The Task object has not been assigned and is a **recurring task**.
0x00000502	The Task object is the **task assignee's** copy of the Task object.
0x00000503	The Task object is the **task assigner's** copy of the Task object.
0x00000500	None of the other conditions apply.

2.2.2.2 Task Object Specific Properties

2.2.2.2.1 PidLidTaskMode Property

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

The **PidLidTaskMode** property ([MS-OXPROPS] section 2.325) on a Task object has no meaning and is set to zero. For details about the **PidLidTaskMode** property on a task communication, see section 2.2.3.2.

2.2.2.2.2 PidLidTaskStatus Property

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

The **PidLidTaskStatus** property ([MS-OXPROPS] section 2.336) specifies the status of the user's progress on the task. The value is one of the following.

Value	Meaning
0x00000000	The user has not started work on the Task object. If the property is set to this value, the value of the **PidLidPercentComplete** property (section 2.2.2.2.3) MUST be 0.0.
0x00000001	The user's work on this Task object is in progress. If the property is set to this value, the value of the **PidLidPercentComplete** property MUST be greater than 0.0 and less than 1.0.
0x00000002	The user's work on this Task object is complete. If the property is set to this value, the value of the **PidLidPercentComplete** property MUST be 1.0, the value of the **PidLidTaskDateCompleted** property (section 2.2.2.2.9) MUST be the current date, and the value of the **PidLidTaskComplete** property (section 2.2.2.2.20) MUST be 0x01.
0x00000003	The user is waiting on somebody else.
0x00000004	The user has deferred work on the Task object.

2.2.2.2.3 PidLidPercentComplete Property

Type: **PtypFloating64** ([MS-OXCDATA] section 2.11.1)

The **PidLidPercentComplete** property ([MS-OXPROPS] section 2.202) indicates the progress the user has made on a task. The value MUST be a number greater than or equal to 0.0 and less than or equal to 1.0, where 1.0 indicates that work is completed and 0.0 indicates that work has not begun.

2.2.2.2.4 PidLidTaskStartDate Property

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

The **PidLidTaskStartDate** property ([MS-OXPROPS] section 2.334) specifies the date on which the user expects work on the task to begin. The date is in the user's local time zone. The task has no start date if this property is unset or is set to 0x5AE980E0 (1,525,252,320). If the task has a start date, the value MUST have a time component of 12:00 midnight, and the **PidLidTaskDueDate** property (section 2.2.2.2.5) and the **PidLidCommonStart** property (section 2.2.2.1.3) MUST also be set.

2.2.2.2.5 PidLidTaskDueDate Property

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

The **PidLidTaskDueDate** property ([MS-OXPROPS] section 2.315) specifies the date by which the user expects work on the task to be complete. The date is in the user's local time zone. The task has no due date if this property is unset or is set to 0x5AE980E0 (1,525,252,320). However, a due date is optional only if no start date is indicated in the **PidLidTaskStartDate** property (section 2.2.2.2.4). If the task has a due date, the value MUST have a time component of 12:00 midnight, and the **PidLidCommonEnd** property (section 2.2.2.1.4) MUST also be set. If the **PidLidTaskStartDate** property has a start date, then the value of this property MUST be greater than or equal to the value of the **PidLidTaskStartDate** property.

2.2.2.2.6 PidLidTaskResetReminder Property

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

The **PidLidTaskResetReminder** property ([MS-OXPROPS] section 2.332) indicates whether future recurring tasks need reminders, even though the value of the **PidLidReminderSet** property ([MS-

OXORMDR] section 2.2.1.1) is zero (FALSE). The **PidLidTaskResetReminder** property is set to nonzero (TRUE) when the task's reminder is dismissed, and is set to zero (FALSE) otherwise. If this property is left unset, the value zero (FALSE) is assumed.

As specified in [MS-OXORMDR], the **PidLidReminderSet** property indicates whether a reminder is set on the Task object. However, the **PidLidReminderSet** property indicates only the presence of a reminder on a single Task object. It cannot be used alone to determine whether a future recurring task needs a reminder.

This is best understood by example. Suppose that the user wants reminders for a series of recurring tasks. The client creates a Task object and sets the **PidLidReminderSet** property to nonzero (TRUE). At the appropriate time, the client presents the user with a reminder. When the user dismisses the reminder, the client sets the **PidLidReminderSet** property to zero (FALSE) and sets the **PidLidTaskResetReminder** property to nonzero (TRUE). Later, the user completes the task, and the client creates a new recurring task. As stated, the user wanted the new recurring task to have a reminder, but the last known value of the **PidLidReminderSet** property was zero (FALSE). The client uses the nonzero (TRUE) value of the **PidLidTaskResetReminder** property to determine that the user had set and dismissed a reminder on a previous recurring task. If the value of the **PidLidTaskResetReminder** property had been zero (FALSE), the client would determine that the user had never set a reminder on the task at all. The client sets a new reminder, as specified in [MS-OXORMDR], if the value of either **PidLidReminderSet** or **PidLidTaskResetReminder** is nonzero (TRUE).

2.2.2.2.7 PidLidTaskAccepted Property

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

The **PidLidTaskAccepted** property ([MS-OXPROPS] section 2.307) indicates whether a task assignee has replied to a task request for this Task object. The client sets this property to zero (FALSE) for a new Task object and to nonzero (TRUE) when a Task object is either accepted or rejected. If left unset, a value of zero (FALSE) is assumed.

2.2.2.2.8 PidLidTaskDeadOccurrence Property

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

The **PidLidTaskDeadOccurrence** property ([MS-OXPROPS] section 2.314) indicates whether a new recurring task remains to be generated. The client sets this property to zero (FALSE) on a new Task object. The client sets this property to nonzero (TRUE) when it generates the last recurring task.

2.2.2.2.9 PidLidTaskDateCompleted Property

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

The **PidLidTaskDateCompleted** property ([MS-OXPROPS] section 2.313) specifies the date when the user completed work on the task. This property can be left unset; if set, this property MUST have a time component of 12:00 midnight in the local time zone, converted to UTC.

2.2.2.2.10 PidLidTaskLastUpdate Property

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

The **PidLidTaskLastUpdate** property ([MS-OXPROPS] section 2.323) specifies the date and time of the most recent change made to the Task object. The most recent change is specified by the **PidLidTaskHistory** property (section 2.2.2.2.18). The value is in UTC.

2.2.2.2.11 PidLidTaskActualEffort Property

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

The **PidLidTaskActualEffort** property ([MS-OXPROPS] section 2.308) specifies the number of minutes that the user actually spent working on a task. The value MUST be greater than or equal to zero and less than 0x5AE980DF (1,525,252,319), where 480 minutes equal one day and 2400 minutes equal one week (8 hours in a work day and 5 work days in a work week).

2.2.2.2.12 PidLidTaskEstimatedEffort Property

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

The **PidLidTaskEstimatedEffort** property ([MS-OXPROPS] section 2.316) specifies the number of minutes that the user expects to work on a task. The value MUST be greater than or equal to zero and less than 0x5AE980DF (1,525,252,319), where 480 minutes equal one day and 2400 minutes equal one week (8 hours in a work day and 5 work days in a work week).

2.2.2.2.13 PidLidTaskVersion Property

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

The **PidLidTaskVersion** property ([MS-OXPROPS] section 2.339) indicates which copy is the latest update of a Task object. An update with a lower version than the Task object is ignored. When embedding a Task object in a task communication, the client sets the current version of the embedded Task object on the task communication as well.

The initial value of this property is 1. The value is incremented only when the Task object is owned by the user. The Task object is owned by the user when the **PidLidTaskState** property (section 2.2.2.2.14) is set to 0x00000001, 0x00000002, or 0x00000004.

2.2.2.2.14 PidLidTaskState Property

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

The **PidLidTaskState** property ([MS-OXPROPS] section 2.335) indicates the current assignment state of the Task object. The value is one of the following.

Value	Meaning
0x00000001	The Task object is not assigned.
0x00000002	The Task object is the task assignee's copy of an assigned Task object.
0x00000003	The Task object is the task assigner's copy of an assigned Task object.
0x00000004	The Task object is the task assigner's copy of a rejected Task object.
0x00000000	This Task object was created to correspond to a Task object that was embedded in a task rejection but could not be found locally.

2.2.2.2.15 PidLidTaskRecurrence Property

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

The **PidLidTaskRecurrence** property ([MS-OXPROPS] section 2.331) contains a **RecurrencePattern** structure, as specified in [MS-OXOCAL] section 2.2.1.44.1, that provides information about recurring tasks. Both the **DeletedInstanceCount** field and the **ModifiedInstanceCount** field of the **RecurrencePattern** structure MUST be set zero.

2.2.2.2.16 PidLidTaskAssigners Property

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

The **PidLidTaskAssigners** property ([MS-OXPROPS] section 2.310) contains a stack of entries, each representing a task assigner. The most recent task assigner (that is, the top of the stack) appears at the end.

The format of this property is shown in the following table. Unless otherwise indicated, the types are specified in ([MS-OXCDATA] section 2.11.1.

Size in bytes	Type	Name	Description
4	**PtypInteger32**	*cAssigners*	Number of task assigners.
4	**PtypInteger32**	*cbAssigner*	Size of the task assigner data to follow, in bytes.
Variable	**Address Book EntryID** structure ([MS-OXCDATA] section 2.2.5.2)	*EID*	A structure that represents the task assigner.
Variable	**PtypString8**	*szDisplayName*	The task assigner's **display name**, using multibyte characters.
Variable	**PtypString**	*wzDisplayName*	The task assigner's display name, using **Unicode** characters.
Next task assigner's data begins here.			

2.2.2.2.17 PidLidTaskStatusOnComplete Property

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

The **PidLidTaskStatusOnComplete** property ([MS-OXPROPS] section 2.337) indicates whether the task assignee has been requested to send an e-mail status report, which is an e-mail message with "Task Completed" as the subject, when the task assignee completes the assigned task. The client sets this property to nonzero (TRUE) when the task assignee has been requested to send an e-mail status report; otherwise, this property is set to zero (FALSE).

2.2.2.2.18 PidLidTaskHistory Property

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

The **PidLidTaskHistory** property ([MS-OXPROPS] section 2.321) indicates the type of change that was last made to the Task object. When the value of this property is set, the **PidLidTaskLastUpdate** property (section 2.2.2.2.10) MUST also be set to the current time.

The value is one of the following (listed in order of decreasing priority).

Value	Meaning
0x00000004	The **PidLidTaskDueDate** property (section 2.2.2.2.5) changed.
0x00000003	Another property was changed.
0x00000001	The task assignee accepted this Task object.
0x00000002	The task assignee rejected this Task object.
0x00000005	The Task object has been assigned to a task assignee.
0x00000000	No changes were made.

2.2.2.2.19 PidLidTaskUpdates Property

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

The **PidLidTaskUpdates** property ([MS-OXPROPS] section 2.338) indicates whether the task assignee has been requested to send a task update when the assigned Task object changes. The client sets this property to nonzero (TRUE) when the task assignee has been requested to send a task update; otherwise, this property is set to zero (FALSE).

2.2.2.2.20 PidLidTaskComplete Property

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

The **PidLidTaskComplete** property ([MS-OXPROPS] section 2.311) indicates whether the task has been completed. The client sets this property to nonzero (TRUE) when the task has been completed; otherwise, this property is set to zero (FALSE).

2.2.2.2.21 PidLidTaskFCreator Property

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

The **PidLidTaskFCreator** property ([MS-OXPROPS] section 2.317) indicates that the Task object was originally created by the action of the current user or user agent instead of by the processing of a task request. The client sets this property to nonzero (TRUE) when the user creates the task and to zero (FALSE) when the task was assigned by another user. If this property is left unset, a value of nonzero (TRUE) is assumed.

2.2.2.2.22 PidLidTaskOwner Property

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

The **PidLidTaskOwner** property ([MS-OXPROPS] section 2.329) specifies the name of the **task owner**.

2.2.2.2.23 PidLidTaskMultipleRecipients Property

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

The **PidLidTaskMultipleRecipients** property ([MS-OXPROPS] section 2.326) specifies optimization hints about the recipients of a Task object.

This property can be left unset; if set, it MUST be set to a bitwise OR of zero or more of the following flags.

Flag name	Value	Meaning
Sent	0x00000001	The Task object has multiple **primary recipients**.
Received	0x00000002	Although the "Sent" hint was not present, the client detected that the Task object has multiple primary recipients.
Reserved	0x00000004	This value has no effect. The client MAY<1> set this flag to indicate that the Task object is a team task.

2.2.2.2.24 PidLidTaskAssigner Property

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

The **PidLidTaskAssigner** property ([MS-OXPROPS] section 2.309) specifies the name of the user that last assigned the task. If the task has not been assigned, this property is left unset.

Because this property is set by the client after the task assignee receives a task request, the property will not be set on the task assigner's copy of the Task object.

When the client adds or removes a task assigner from the stack of task assigners listed in the **PidLidTaskAssigners** property (section 2.2.2.2.16), this property is set to the added or removed task assigner.

2.2.2.2.25 PidLidTaskLastUser Property

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

The **PidLidTaskLastUser** property ([MS-OXPROPS] section 2.324) specifies the name of the most recent user to have been the task owner.

Before a client sends a task request, it sets this property to the name of the task assigner.

Before a client sends a task acceptance, it sets this property to the name of the task assignee.

Before a client sends a task rejection, it sets this property to the name of the task assigner.

2.2.2.2.26 PidLidTaskOrdinal Property

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

The **PidLidTaskOrdinal** property ([MS-OXPROPS] section 2.328) specifies a number that aids custom sorting of Task objects. This property can be left unset; if set, its value MUST be greater than 0x800186A0 (-2,147,383,648) and less than 0x7FFE7960 (2,147,383,648) and MUST be unique among Task objects in the same folder.

Whenever the client sets this property to a number less than the negative of the current value of the **PidTagOrdinalMost** property (section 2.2.1.1) of the folder, the client MUST also update the **PidTagOrdinalMost** property on the folder.

2.2.2.2.27 PidLidTaskLastDelegate Property

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

The **PidLidTaskLastDelegate** property ([MS-OXPROPS] section 2.322) specifies the name of the mailbox's **delegate** who most recently assigned the task. This property contains an empty string if there is no delegate. For details about delegates, see [MS-OXODLGT].

2.2.2.2.28 PidLidTaskFRecurring Property

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

The **PidLidTaskFRecurring** property ([MS-OXPROPS] section 2.319) indicates whether the task includes a recurrence pattern. If this property is unset or is set to zero (FALSE), the task does not include a recurrence pattern. If set to nonzero (TRUE), the **PidLidTaskRecurrence** (section 2.2.2.2.15) and **PidLidTaskDeadOccurrence** (section 2.2.2.2.8) properties MUST also be set.

2.2.2.2.29 PidLidTaskOwnership Property

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

The **PidLidTaskOwnership** property ([MS-OXPROPS] section 2.330) indicates the role of the current user relative to the Task object. The value is one of the following.

Value	Meaning
0x00000000	The Task object is not assigned.
0x00000001	The Task object is the task assigner's copy of the Task object.
0x00000002	The Task object is the task assignee's copy of the Task object.

2.2.2.2.30 PidLidTaskAcceptanceState Property

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

The **PidLidTaskAcceptanceState** property ([MS-OXPROPS] section 2.306) indicates the acceptance state of the task. The value is one of the following.

Value	Meaning
0x00000000	The Task object is not assigned.
0x00000001	The Task object's acceptance status is unknown.
0x00000002	The task assignee has accepted the Task object. This value is set when the client processes a task acceptance.
0x00000003	The task assignee has rejected the Task object. This value is set when the client processes a task rejection.

2.2.2.2.31 PidLidTaskFFixOffline Property

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

The **PidLidTaskFFixOffline** property ([MS-OXPROPS] section 2.318) indicates whether the value of the **PidLidTaskOwner** property (section 2.2.2.2.22) is correct. The value zero (FALSE) indicates that the value of the **PidLidTaskOwner** property is correct. A nonzero value (TRUE) indicates that the client cannot determine an accurate value for the **PidLidTaskOwner** property.

When the client sets this property to a nonzero (TRUE) value, the client can also set the **PidLidTaskOwner** property to a generic owner name, such as "Unknown". When the client changes the value of the **PidLidTaskOwner** property, the client updates the **PidLidTaskFFixOffline** property accordingly.

2.2.2.2.32 PidLidTaskGlobalId Property

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

The **PidLidTaskGlobalId** property ([MS-OXPROPS] section 2.320) specifies a unique **GUID** for this task, used to locate an existing task upon receipt of a task response or task update. This property MUST be set for assigned tasks, but it can be left unset for unassigned tasks.

2.2.2.2.33 PidLidTaskCustomFlags Property

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

The **PidLidTaskCustomFlags** property ([MS-OXPROPS] section 2.312) is not used. The client can set this property, but it has no impact on the Task-Related Objects Protocol and is ignored by the server.

2.2.2.2.34 PidLidTaskRole Property

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

The **PidLidTaskRole** property ([MS-OXPROPS] section 2.333) is not used. The client can set this property, but it has no impact on the Task-Related Objects Protocol and is ignored by the server.

2.2.2.2.35 PidLidTaskNoCompute Property

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

The **PidLidTaskNoCompute** property ([MS-OXPROPS] section 2.327) is not used. The client can set this property, but it has no impact on the Task-Related Objects Protocol and is ignored by the server.

2.2.2.2.36 PidLidTeamTask Property

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

The **PidLidTeamTask** property ([MS-OXPROPS] section 2.340) is not used. The client can set this property, but it has no impact on the Task-Related Objects Protocol and is ignored by the server.

2.2.3 Task Communications Properties

The property requirements specified in sections 2.2.3.1 through 2.2.3.3.2 are specific to task requests, task acceptances, task rejections, and task updates (collectively, task communications).

2.2.3.1 PidTagProcessed Property

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

The **PidTagProcessed** property ([MS-OXPROPS] section 2.942) indicates whether a client has already processed a received a task communication. This property is left unset until processing has completed and then is set to nonzero (TRUE).

2.2.3.2 PidLidTaskMode Property

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

The **PidLidTaskMode** property ([MS-OXPROPS] section 2.325) specifies the assignment status of the Task object that is embedded in the task communication. The following table specifies the valid values for this property.

Value	Meaning
0	The Task object is not assigned.
1	The Task object is embedded in a task request.
2	The Task object has been accepted by the task assignee.
3	The Task object was rejected by the task assignee.
4	The Task object is embedded in a task update.
5	The Task object was assigned to the task assigner (self-delegation).

2.2.3.3 Additional Property Constraints

In some cases, the task communication has specific requirements for properties that are otherwise inherited. This section specifies these specific requirements.

2.2.3.3.1 PidTagMessageClass Property

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

The **PidTagMessageClass** property ([MS-OXCMSG] section 2.2.1.3) specifies the type of the Message object. The value is one of the following strings and is case-insensitive.

String	Type of task communication
"IPM.TaskRequest"	Task request
"IPM.TaskRequest.Accept"	Task acceptance
"IPM.TaskRequest.Decline"	Task rejection
"IPM.TaskRequest.Update"	Task update

2.2.3.3.2 PidTagIconIndex Property

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

The **PidTagIconIndex** property ([MS-OXOMSG] section 2.2.1.10) specifies which icon is to be used by a user interface to represent the task communication. If this property exists, its value is a hint to the client's user interface. The user interface can ignore the value and use another method of determining which icon to display to the user.

The value is one of the following.

Value	Type of task communication
0x00000504	Task request
0x00000505	Task acceptance
0x00000506	Task rejection
0x00000500	Task update
0xFFFFFFFF	Unspecified

3 Protocol Details

3.1 Client Details

The client creates and manipulates a Task object and a task communication and in all other ways operates within the client role as specified in [MS-OXCMSG].

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.

This protocol uses the abstract data model that is specified in [MS-OXCMSG] section 3.1.1 with the following adaptations:

- The Task object and the task communications are extensions of the Message object.
- A Task object is created in the Tasks folder, which is a Folder object, unless the end user or user agent explicitly specifies another folder.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Creating a Task Object and a Task Communication

To create a Task object, the client creates a Message object as specified in [MS-OXCMSG]. The client adds a user as a **carbon copy (Cc) recipient** if that user is to receive task updates. The client adds a user as a **blind carbon copy (Bcc) recipient** and sets the **PidLidTaskStatusOnComplete** property (section 2.2.2.2.17) to nonzero (TRUE) if that user is to receive an e-mail status report when the task is completed. The client sets properties in accordance with the requirements in section 2.2.2 of this document and saves the Message object as specified in [MS-OXCMSG].

Although Task objects support recipients, the client does not submit a Task object to the server for delivery to other users. Instead, the client submits a task communication. To create a task communication, the client creates a Message object as specified in [MS-OXCMSG], sets properties in accordance with the requirements in section 2.2.3 of this document, and saves the Message object as specified in [MS-OXCMSG]. The client embeds a copy of the Task object as an **Attachment object** within the task communication (the embedding object). For details about embedding a Task object, see section 3.1.4.3. A Task object that is in a **public folder** is not assigned. Therefore, the client MUST NOT create a task request for a Task object that is in a public folder.

3.1.4.2 Modifying a Task Object and a Task Communication

To modify a Task object or a task communication, the client opens a Message object as specified in [MS-OXCMSG], modifies any of the properties in accordance with the requirements in section 2.2.2 and section 2.2.3 of this document, and saves the Message object as specified in [MS-OXCMSG]. The updated Task object is embedded in the task update as specified in section 3.1.4.3.

When the client changes a Task object, it sends a task update to all of the Cc recipients. For details about sending a task update, see section 3.1.4.4.

When the client marks a Task object as complete by setting the **PidLidTaskStatus** property (section 2.2.2.2.2), it sends an e-mail message with "Task Completed" as the subject to all of the Bcc recipients if the value of the **PidLidTaskStatusOnComplete** property (section 2.2.2.2.17) is nonzero (TRUE).

3.1.4.3 Embedding a Task Object

Before a task can be assigned, the client embeds a copy of the Task object as an Attachment object within the task communication (the embedding object). To embed a Task object, the client MUST complete the following steps in the order specified:

1. Create an Attachment object on the embedding object, as specified in [MS-OXCMSG] section 3.1.4.18. This Attachment object MUST be the first Attachment object created on the embedding object.
2. Set the **PidTagAttachMethod** property ([MS-OXCMSG] section 2.2.2.9) to afEmbeddedMessage (0x00000005), the **PidTagRenderingPosition** property ([MS-OXCMSG] section 2.2.2.16) to 0xFFFFFFFF, and the **PidTagAttachmentHidden** property ([MS-OXCMSG] section 2.2.2.24) to 0x01, as specified in [MS-OXCMSG].<2>
3. Open the Attachment object as an **Embedded Message object**, as specified in [MS-OXCMSG] section 3.1.4.16.
4. Set the appropriate properties of the Embedded Message object (the embedded Task object) as specified throughout this document.
5. If the original Task object has a **PidLidTaskGlobalId** property (section 2.2.2.2.32), copy it to the embedded Task object. Otherwise, set the value of the **PidLidTaskGlobalId** property of the embedded Task object to a new, unique GUID.
6. Save the Embedded Message object, as specified in [MS-OXCMSG] section 3.1.4.19.
7. Release the Message object by using the **RopRelease** **remote operation (ROP)** ([MS-OXCROPS] section 2.2.15.3).
8. Release the Attachment object by using the **RopRelease** ROP.

3.1.4.4 Sending a Task Communication

The client creates a task communication, as specified in section 3.1.4.1, and then adds the recipients and saves the Message object, as specified in [MS-OXCMSG]. The client then performs other actions, which are dependent on the particular type of task communication that will be sent. To send the task communication, the client submits the task communication for delivery, as specified in [MS-OXOMSG].

Task request: Before the client sends a task request, it computes the name of the new owner of the task by retrieving the primary recipients from the Task object. If there is only one primary recipient,

its display name is the name of the new owner. If there are multiple primary recipients, the new owner name is derived by concatenating the display names of all the primary recipients, separated with semicolons (";"). A task can be assigned to only one task assignee. If a Task object has more than one primary recipient, the task is shared, not assigned. A Task object that is in a public folder is not assigned. The client sets the value of the **PidLidTaskOwner** property (section 2.2.2.2.22) of the Task object with this new owner name. The client also sets the value of the **PidLidTaskGlobalId** property (section 2.2.2.2.32) of the Task object to a new, unique GUID if it does not already have one.

Task acceptance: When the client sends a task acceptance, the client creates a local Task object and copies to it the relevant properties from the embedded Task object of the task request. Then, the client sets the recipient to the last task assigner listed in the **PidLidTaskAssigners** property (section 2.2.2.2.16) of the Task object. The stack of task assigners is deleted from the **PidLidTaskAssigners** property, leaving only the most recent task assigner. The client sets the **PidLidTaskOwnership** property (section 2.2.2.2.29) to 0x00000002. The client does not send a task acceptance when the Task object has more than one primary recipient.

Task rejection: When the client sends a task rejection, it removes the last entry from the **PidLidTaskAssigners** property of the Task object. The client sets the value of the **PidLidTaskOwner** property of the Task object to the name from this last entry. Then, the client sets the recipient to the last task assigner listed in the **PidLidTaskAssigners** property of the Task object. The client does not send a task rejection when the Task object has more than one primary recipient.

Task update: When the client changes a Task object, it sends a task update to all Cc recipients of the Task object if the **PidLidTaskUpdates** property (section 2.2.2.2.19) is set to nonzero (TRUE). The revised Task object is embedded in the task update, as specified in section 3.1.4.3. When the client sends a task update, it sets the recipient to the last task assigner listed in the **PidLidTaskAssigners** property of the Task object. The client does not send a task update when the Task object has more than one primary recipient.

3.1.4.5 Receiving a Task Communication

When the client receives a task communication, the client's handling of the task communication depends on the type of task communication that is received.

Task request: When the client receives a task request, it appends an entry that represents the sender of the task request to the **PidLidTaskAssigners** property (section 2.2.2.2.16) of the Task object and sets the value of the **PidLidTaskOwner** property (section 2.2.2.2.22) of the Task object to the name of the task assignee. The client also adds the sender to the Bcc recipients of the Task object if the value of the **PidLidTaskUpdates** property (section 2.2.2.2.19) of the Task object is nonzero.

Task acceptance, task rejection, or task update: A task acceptance, task rejection, or task update contains an embedded Task object that is an update to the client's local Task object. When the client receives a task acceptance, task rejection, or task update, the client locates the local Task object by using the **PidLidTaskGlobalId** property (section 2.2.2.2.32) of the embedded Task object along with a **restriction (2)**. For details about using a restriction (2) to find a Message object, see [MS-OXCTABL]. If the client can locate the local Task object, it copies any relevant properties from the embedded Task object to the local Task object. If the client cannot locate the local Task object, the client SHOULD<3> create one.

3.1.4.6 Generating Instances of Recurring Tasks

The client does not generate all instances of a recurring task at once. It begins by generating an initial instance only. In many cases, this instance will already exist when a recurrence pattern is added to it.

3.1.4.6.1 Determining Whether to Generate a New Instance

The client determines whether to generate a new instance of the recurring task when the prior instance (a) is completed — that is, the **PidLidTaskStatus** property (section 2.2.2.2.2) is marked as Complete; (b) is deleted; or (c) is given a new recurring start date or due date.

While determining whether to generate a new instance of a recurring task, the client does not generate a new instance if the value of the **PidLidTaskFRecurring** property (section 2.2.2.2.28) is 0x00 or if the value of the **PidLidTaskDeadOccurrence** property (section 2.2.2.2.8) is 0x01.

The client also considers the criteria specified in the recurrence pattern. For details about recurrence patterns, see [MS-OXOCAL]. If the recurrence pattern specifies a valid end date and a positive count of occurrences, the client decrements the count of occurrences, saves the new recurrence pattern, and generates a new instance. If the occurrence count reaches 0, the client sets the value of the **PidLidTaskDeadOccurrence** property to 0x01.

3.1.4.6.2 New Instance Dates

Some recurrence patterns are sliding. In such cases, the recurrence pattern does not specify the absolute date of each occurrence. Rather, the recurrence pattern specifies a date that is relative to the completion date of the prior instance. The client computes the date of the new instance accordingly.

Having determined from the recurrence pattern the appropriate date for a new instance, the client determines and sets the values for the start date and due date properties of the new instance. The new values of these properties are determined by combining the values of these properties from the prior instance with the newly calculated instance date, as follows: If the prior instance does not have a start date, the new instance does not have a start date, and the new due date is the newly calculated instance date. Otherwise, the new start date is the newly calculated instance date and the new instance and the new due date is the sum of the new start date and the difference between the old due date and the old start date. In other words, new due date = new start date + (old due date - old start date).

Finally, the client sets the reminder properties of the new instance, as specified in [MS-OXORMDR]. In particular, the client sets the **PidLidReminderSet** property ([MS-OXORMDR] section 2.2.1.1) of the new instance to 0x01 if the reminder time has not already passed and (a) the **PidLidReminderSet** property of the prior instance is 0x01, or (b) the **PidLidTaskResetReminder** property (section 2.2.2.2.6) of the prior instance is 0x01. If the reminder time has already passed but either condition (a) or (b) applies, the client sets **PidLidTaskResetReminder** to 0x01 so that future instances can continue to follow the same logic.

3.1.4.6.3 Archive Instances

If a new instance is warranted, the client does not create a new Task object for the new instance. A new Task object would have distinct values for properties such as **PidTagSearchKey** ([MS-OXCPRPT] section 2.2.1.9), **PidTagEntryId** ([MS-OXCPERM] section 2.2.4), and others that might affect later efforts to locate and identify the Task object. Instead, the client updates the properties of the existing Task object and uses it as the new instance. If preferred, the client first creates a new Task object to represent the now-completed task.

To create a Task object to represent the now-completed task, the client creates a new Task object, as usual. Then, the client copies any relevant recipients, attachments, and properties, as specified in [MS-OXCMSG], from the prior Task object to the new Task object, with these exceptions.

Property	Action
PidLidTaskOwnership (section 2.2.2.2.29)	Set to 0, not assigned.
PidLidTaskAcceptanceState (section 2.2.2.2.30)	Set to 0, not assigned.
PidLidTaskState (section 2.2.2.2.14)	Set to 1, not assigned.
PidLidTaskMode (section 2.2.2.2.1)	Set to 0.
PidLidTaskOrdinal (section 2.2.2.2.26)	Set to a unique ordinal.
PidTagReadReceiptRequested ([MS-OXOMSG] section 2.2.1.28)	Set to 0x00.
PidTagOriginatorDeliveryReportRequested ([MS-OXOMSG] section 2.2.1.20)	Set to 0x00.
PidLidTaskAssigner (section 2.2.2.2.24)	Set to "", empty string.
PidTagSenderName ([MS-OXOMSG] section 2.2.1.48)	Delete.
PidTagSenderEmailAddress ([MS-OXOMSG] section 2.2.1.46)	Delete.
PidTagSenderAddressType ([MS-OXOMSG] section 2.2.1.45)	Delete.
PidTagSenderEntryId ([MS-OXOMSG] section 2.2.1.47)	Delete.
PidTagSenderSearchKey ([MS-OXOMSG] section 2.2.1.49)	Delete.
PidTagSentRepresentingName ([MS-OXOMSG] section 2.2.1.54)	Delete.
PidTagSentRepresentingEmailAddress ([MS-OXOMSG] section 2.2.1.52)	Delete.
PidTagSentRepresentingAddressType ([MS-OXOMSG] section 2.2.1.51)	Delete.
PidTagSentRepresentingEntryId ([MS-OXOMSG] section 2.2.1.53)	Delete.
PidTagSentRepresentingSearchKey ([MS-OXOMSG] section 2.2.1.55)	Delete.
PidLidTaskAssigners (section 2.2.2.2.16)	Delete.
PidLidTaskFFixOffline (section 2.2.2.2.31)	Set to 0x00.
PidLidTaskDeadOccurrence (section 2.2.2.2.8)	Set to 0x01.
PidLidTaskStatus (section 2.2.2.2.2)	Set to 2, complete.
PidLidTaskComplete (section 2.2.2.2.20)	Set to 0x01.
PidLidPercentComplete (section 2.2.2.2.3)	Set to 1.0, 100%.

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

The server processes a client's requests regarding a Task object and a task communication and in all other ways operates within the server role as specified in [MS-OXCMSG].

3.2.1 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.

This protocol uses the abstract data model that is specified in [MS-OXCMSG] section 3.2.1 with the following adaptations:

- The Task object and the task communications are extensions of the Message object.
- A Task object is created in the Tasks folder, which is a Folder object, unless the end user or user agent explicitly specifies another folder.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The server responds to client requests as specified in [MS-OXCMSG] section 3.2.5.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

The examples in sections 4.1 and 4.2 use both **named properties** and **tagged properties**. The **property ID** of a named property is provided by the server. Therefore, before setting or reading any properties of a Task object, the client asks the server to perform a mapping from **property names** or **long IDs (LIDs)** to property IDs by using the **RopGetPropertyIdsFromNames** ROP ([MS-OXCROPS] section 2.2.8.1).

The following table lists all of the named properties that are used in the examples.

Property	Property set GUID	LID
PidLidTaskComplete (section 2.2.2.2.20)	{00062003-0000-0000-C000-000000000046}	0x0000811C
PidLidTaskStatus (section 2.2.2.2.2)	{00062003-0000-0000-C000-000000000046}	0x00008101
PidLidPercentComplete (section 2.2.2.2.3)	{00062003-0000-0000-C000-000000000046}	0x00008102
PidLidTaskActualEffort (section 2.2.2.2.11)	{00062003-0000-0000-C000-000000000046}	0x00008110
PidLidTaskEstimatedEffort (section 2.2.2.2.12)	{00062003-0000-0000-C000-000000000046}	0x00008111
PidLidTaskUpdates (section 2.2.2.2.19)	{00062003-0000-0000-C000-000000000046}	0x0000811B
PidLidTaskStatusOnComplete (section 2.2.2.2.17)	{00062003-0000-0000-C000-000000000046}	0x00008119
PidLidTaskFFixOffline (section 2.2.2.2.31)	{00062003-0000-0000-C000-000000000046}	0x0000812C
PidLidTaskOwnership (section 2.2.2.2.29)	{00062003-0000-0000-C000-000000000046}	0x00008129
PidLidTaskAcceptanceState (section 2.2.2.2.30)	{00062003-0000-0000-C000-000000000046}	0x0000812A
PidLidTaskState (section 2.2.2.2.14)	{00062003-0000-0000-C000-000000000046}	0x00008113
PidLidTaskOrdinal (section 2.2.2.2.26)	{00062003-0000-0000-C000-000000000046}	0x00008123
PidLidTaskHistory (section 2.2.2.2.18)	{00062003-0000-0000-C000-000000000046}	0x0000811A
PidLidTaskLastUpdate (section 2.2.2.2.10)	{00062003-0000-0000-C000-000000000046}	0x00008115
PidLidTaskLastUser (section 2.2.2.2.25)	{00062003-0000-0000-C000-000000000046}	0x00008122
PidLidTaskLastDelegate (section 2.2.2.2.27)	{00062003-0000-0000-C000-000000000046}	0x00008125

Property	Property set GUID	LID
PidLidTaskVersion (section 2.2.2.2.13)	{00062003-0000-0000-C000-000000000046}	0x00008112
PidLidTaskOwner (section 2.2.2.2.22)	{00062003-0000-0000-C000-000000000046}	0x0000811F
PidLidTaskFRecurring (section 2.2.2.2.28)	{00062003-0000-0000-C000-000000000046}	0x00008126
PidLidTaskMode (section 2.2.3.2)	{00062008-0000-0000-C000-000000000046}	0x00008518
PidLidTaskGlobalId (section 2.2.2.2.32)	{00062008-0000-0000-C000-000000000046}	0x00008519
PidLidTaskDueDate (section 2.2.2.2.5)	{00062003-0000-0000-C000-000000000046}	0x00008105
PidLidTaskStartDate (section 2.2.2.2.5)	{00062003-0000-0000-C000-000000000046}	0x00008104
PidLidTaskDateCompleted (section 2.2.2.2.9)	{00062003-0000-0000-C000-000000000046}	0x0000810F
PidLidTaskAccepted (section 2.2.2.2.7)	{00062003-0000-0000-C000-000000000046}	0x00008108
PidLidTaskResetReminder (section 2.2.2.2.6)	{00062003-0000-0000-C000-000000000046}	0x00008107
PidLidTaskMultipleRecipients (section 2.2.2.2.23)	{00062003-0000-0000-C000-000000000046}	0x00008120
PidLidTaskDeadOccurrence (section 2.2.2.2.8)	{00062003-0000-0000-C000-000000000046}	0x00008109
PidLidTaskRole (section 2.2.2.2.34)	{00062003-0000-0000-C000-000000000046}	0x00008127
PidLidTaskAssigners (section 2.2.2.2.16)	{00062003-0000-0000-C000-000000000046}	0x00008117
PidLidTaskRecurrence (section 2.2.2.2.15)	{00062003-0000-0000-C000-000000000046}	0x00008116
PidLidTaskAssigner (section 2.2.2.2.24)	{00062003-0000-0000-C000-000000000046}	0x00008121
PidLidTaskFCreator (section 2.2.2.2.21)	{00062003-0000-0000-C000-000000000046}	0x0000811E
PidLidCommonStart (section 2.2.2.1.3)	{00062008-0000-0000-C000-000000000046}	0x00008516
PidLidCommonEnd (section 2.2.2.1.4)	{00062008-0000-0000-C000-000000000046}	0x00008517

The server might respond with the following property IDs, which will be used in the examples (the actual property IDs are at the discretion of the server).

Property	Property ID
PidLidTaskComplete	0x8149
PidLidTaskStatus	0x8146
PidLidPercentComplete	0x8147
PidLidTaskActualEffort	0x814D
PidLidTaskEstimatedEffort	0x814E
PidLidTaskUpdates	0x82C3
PidLidTaskStatusOnComplete	0x82C4
PidLidTaskFFixOffline	0x8156
PidLidTaskOwnership	0x8154
PidLidTaskAcceptanceState	0x8151
PidLidTaskState	0x8148
PidLidTaskOrdinal	0x815D
PidLidTaskHistory	0x8150
PidLidTaskLastUpdate	0x8153
PidLidTaskLastUser	0x8152
PidLidTaskLastDelegate	0x82C5
PidLidTaskVersion	0x8158
PidLidTaskOwner	0x801B
PidLidTaskFRecurring	0x814B
PidLidTaskMode	0x8212
PidLidTaskGlobalId	0x8211
PidLidTaskDueDate	0x8145
PidLidTaskStartDate	0x8144
PidLidTaskDateCompleted	0x814A
PidLidTaskAccepted	0x82C2
PidLidTaskResetReminder	0x815C
PidLidTaskMultipleRecipients	0x814F
PidLidTaskDeadOccurrence	0x814C
PidLidTaskRole	0x8157
PidLidTaskAssigners	0x82C8

Property	Property ID
PidLidTaskRecurrence	0x815B
PidLidTaskAssigner	0x8159
PidLidTaskFCreator	0x82CA
PidLidCommonStart	0x81BD
PidLidCommonEnd	0x81BC

4.1 Sending a Task Request

Mary Kay Andersen assigns a task to her coworker, Paul West. The following is a description of what a client might do to accomplish Mary's intentions.

The client begins by obtaining property IDs from the server, as described in section 4.

To create the task request, the client uses the **RopCreateMessage** ROP ([MS-OXCROPS] section 2.2.6.2). The server returns a success code and a **handle** to a Message object. The client uses the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) to transmit Mary's data to the server.

The property types in the following tables are described in [MS-OXCDATA] section 2.11.1.

Property	Property ID	Type	Value
PidTagMessageClass (section 2.2.3.3.1)	0x001A	0x001F (**PtypString**)	"IPM.TaskRequest"
PidTagIconIndex (section 2.2.3.3.2)	0x1080	0x0003 (**PtypInteger32**)	0xFFFFFFFF
PidLidTaskMode (section 2.2.2.2.1)	0x8212	0x0003 (**PtypInteger32**)	0x00000001

The client provides the actual Task object in an Embedded Message object. The protocol creates an Attachment object into which it will embed the Task object by using the **RopCreateAttachment** ROP ([MS-OXCROPS] section 2.2.6.13), which returns a handle to the new Attachment object. The client then uses this handle with the **RopSetProperties** ROP to set the **PidTagAttachMethod** property ([MS-OXPROPS] section 2.671) to afEmbeddedMessage (0x00000005).

Property	Property ID	Type	Value
PidTagAttachMethod ([MS-OXCMSG] section 2.2.2.9)	0x3705	0x0003 (**PtypInteger32**)	0x00000005 (afEmbeddedMessage)
PidTagRenderingPosition ([MS-OXCMSG] section 2.2.2.16)	0x370B	0x0003 (**PtypInteger32**)	0xFFFFFFFF
PidTagAttachmentHidden ([MS-OXCMSG] section 2.2.2.24)	0x7FFE	0x000B (**PtypBoolean**)	0x01

The client acquires the handle to the Embedded Message object within the Attachment object by using the **RopOpenEmbeddedMessage** ROP ([MS-OXCROPS] section 2.2.6.16), which can be used

as a Task object. The client sets the properties that it wants for this Task object or copies them from a local Task object by using the **RopSetProperties** ROP.

Property	Property ID	Type	Value
PidTagMessageClass	0x001A	0x001F (**PtypString**)	"IPM.Task"
PidTagIconIndex	0x1080	0x0003 (**PtypInteger32**)	0x00000500 (assigner's task)
PidLidTaskComplete (section 2.2.2.2.20)	0x8149	0x000B (**PtypBoolean**)	0x00
PidLidPercentComplete (section 2.2.2.2.3)	0x8147	0x0005 (**PtypFloating64**)	0.0
PidLidTaskStatus (section 2.2.2.2.2)	0x8146	0x0003 (**PtypInteger32**)	0 (Not started)
PidLidTaskActualEffort (section 2.2.2.2.11)	0x814D	0x0003 (**PtypInteger32**)	0
PidLidTaskEstimatedEffort (section 2.2.2.2.12)	0x814E	0x0003 (**PtypInteger32**)	0
PidLidTaskUpdates (section 2.2.2.2.19)	0x82C3	0x000B (**PtypBoolean**)	0x01
PidLidTaskStatusOnComplete (section 2.2.2.2.17)	0x82C4	0x000B (**PtypBoolean**)	0x01
PidLidTaskFFixOffline (section 2.2.2.2.31)	0x8156	0x000B (**PtypBoolean**)	0x00
PidLidTaskOwnership (section 2.2.2.2.29)	0x8154	0x0003 (**PtypInteger32**)	1 (assigner's copy)
PidLidTaskAcceptanceState (section 2.2.2.2.30)	0x8151	0x0003 (**PtypInteger32**)	1 (unknown)
PidLidTaskState (section 2.2.2.2.14)	0x8148	0x0003 (**PtypInteger32**)	3 (assigner's copy)
PidLidTaskOrdinal (section 2.2.2.2.26)	0x815D	0x0003 (**PtypInteger32**)	-1000
PidLidTaskHistory (section 2.2.2.2.18)	0x8150	0x0003 (**PtypInteger32**)	0x00000005 (sent with a task request)
PidLidTaskLastUpdate (section 2.2.2.2.10)	0x8153	0x0040 (**PtypTime**)	2008/02/19 07:00
PidLidTaskLastUser (section 2.2.2.2.25)	0x8152	0x001F (**PtypString**)	"Mary Kay Andersen"
PidLidTaskLastDelegate (section 2.2.2.2.27)	0x82C5	0x001F (**PtypString**)	"Mary Kay Andersen"
PidLidTaskVersion (section	0x8158	0x0003	2

Property	Property ID	Type	Value
2.2.2.2.13)		(**PtypInteger32**)	
PidLidTaskOwner (section 2.2.2.2.22)	0x801B	0x001F (**PtypString**)	"Paul West"
PidLidTaskFRecurring (section 2.2.2.2.28)	0x814B	0x000B (**PtypBoolean**)	0x00
PidLidTaskMode (section 2.2.3.2)	0x8212	0x0003 (**PtypInteger32**)	1 (embedded in a task request)
PidLidTaskGlobalId (section 2.2.2.2.32)	0x8211	0x0102 (**PtypBinary**)	0E B0 1E 03 85 02 EF 4B 9A 14 50 83 B3 BB 4D E9
PidLidTaskMode	0x8212	0x0003 (**PtypInteger32**)	0x00000001

The client then sets other Attachment object properties, as described in [MS-OXCMSG].

The client saves and closes the Embedded Message object by using, in order, the following operations: **RopSaveChangesMessage** ([MS-OXCROPS] section 2.2.6.3) with the handle to the embedded Task object, **RopSaveChangesAttachment** ([MS-OXCROPS] section 2.2.6.15) with the handle to the Attachment object, **RopRelease** ([MS-OXCROPS] section 2.2.15.3) with the handle to the embedded Task object, and **RopRelease** with the handle to the Attachment object.

The client uses the **RopModifyRecipients** ROP ([MS-OXCROPS] section 2.2.6.5) to add Paul's recipient information to the task request.

When Mary is ready to send her task request, the client uses the **RopSetProperties** ROP to commit the properties to the server, the **RopSubmitMessage** ROP ([MS-OXCROPS] section 2.2.7.1) to send it, and then the **RopRelease** ROP to release the task request object.

4.2 Processing a Task Update

Russell King assigned a task to Scott Bishop. Scott updated some of the task properties, such as percent completed, and sent an update. Russell has now received a task update, and Scott's changes need to be merged into his own copy of the task. The following is a description of what a client might do to process the update.

The client begins by obtaining property IDs from the server as described in section 4.

The client obtains a handle to the task update by using the **RopOpenMessage** ROP ([MS-OXCROPS] section 2.2.6.1). The updated task information is part of the Task object that is embedded within the first Attachment object of the task update. To get the attachment, the client uses the handle to the task update with the **RopOpenAttachment** ROP ([MS-OXCROPS] section 2.2.6.12). The client gets a handle to the Embedded Message object from this Attachment object by using the **RopOpenEmbeddedMessage** ROP ([MS-OXCROPS] section 2.2.6.16), which can then be used as the Task object. The client reads properties from the embedded Task object by using the **RopGetPropertiesSpecific** ROP ([MS-OXCROPS] section 2.2.8.3).

The property types in the following tables are described in [MS-OXCDATA] section 2.11.1.

Property	Property ID	Type	Value obtained from server
PidLidTaskStatus (section 2.2.2.2.2)	0x8146	0x0003 (**PtypInteger32**)	0 (Not started)
PidLidPercentComplete (section 2.2.2.2.3)	0x8147	0x0005 (**PtypFloating64**)	0.0 (0%)
PidLidTaskDueDate (section 2.2.2.2.5)	0x8145	0x0040 (**PtypTime**)	<not found>
PidLidTaskStartDate (section 2.2.2.2.5)	0x8144	0x0040 (**PtypTime**)	<not found>
PidLidTaskActualEffort (section 2.2.2.2.11)	0x814D	0x0003 (**PtypInteger32**)	0
PidLidTaskEstimatedEffort (section 2.2.2.2.12)	0x814E	0x0003 (**PtypInteger32**)	0
PidLidTaskDateCompleted (section 2.2.2.2.9)	0x814A	0x0040 (**PtypTime**)	<not found>
PidLidTaskAccepted (section 2.2.2.2.7)	0x82C2	0x000B (**PtypBoolean**)	0x01
PidLidTaskResetReminder (section 2.2.2.2.6)	0x815C	0x000B (**PtypBoolean**)	<not found>
PidLidTaskMultipleRecipients (section 2.2.2.2.23)	0x814F	0x0003 (**PtypInteger32**)	0
PidLidTaskUpdates (section 2.2.2.2.19)	0x82C3	0x000B (**PtypBoolean**)	0x01
PidLidTaskStatusOnComplete (section 2.2.2.2.17)	0x82C4	0x000B (**PtypBoolean**)	0x01
PidLidTaskDeadOccurrence (section 2.2.2.2.8)	0x814C	0x000B (**PtypBoolean**)	<not found>
PidLidTaskComplete (section 2.2.2.2.20)	0x8149	0x000B (**PtypBoolean**)	0x00
PidLidTaskFFixOffline (section 2.2.2.2.31)	0x8156	0x000B (**PtypBoolean**)	0x00
PidLidTaskOwnership (section 2.2.2.2.29)	0x8154	0x0003 (**PtypInteger32**)	2 (task assignee's copy)
PidLidTaskAcceptanceState (section 2.2.2.2.30)	0x8151	0x0003 (**PtypInteger32**)	0 (Not assigned)
PidLidTaskHistory (section 2.2.2.2.18)	0x8150	0x0003 (**PtypInteger32**)	3 (Updated)
PidLidTaskLastUpdate (section 2.2.2.2.10)	0x8153	0x0040 (**PtypTime**)	2008/02/19
PidLidTaskLastUser (section	0x8152	0x001F	"Scott Bishop"

Property	Property ID	Type	Value obtained from server
2.2.2.2.25)		(**PtypString**)	
PidLidTaskLastDelegate (section 2.2.2.2.27)	0x82C5	0x001F (**PtypString**)	"Scott Bishop"
PidLidTaskRole (section 2.2.2.2.34)	0x8157	0x001F (**PtypString**)	""
PidLidTaskVersion (section 2.2.2.2.13)	0x8158	0x0003 (**PtypInteger32**)	4
PidLidTaskState (section 2.2.2.2.14)	0x8148	0x0003 (**PtypInteger32**)	2 (task assignee's copy)
PidLidTaskAssigners (section 2.2.2.2.16)	0x82C8	0x0102 (**PtypBinary**)	<binary data>
PidLidTaskRecurrence (section 2.2.2.2.15)	0x815B	0x0102 (**PtypBinary**)	<not found>
PidLidTaskAssigner (section 2.2.2.2.24)	0x8159	0x001F (**PtypString**)	"Russell King"
PidLidTaskOwner (section 2.2.2.2.22)	0x801B	0x001F (**PtypString**)	"Scott Bishop"
PidLidTaskFCreator (section 2.2.2.2.21)	0x82CA	0x000B (**PtypBoolean**)	0x00
PidLidTaskOrdinal (section 2.2.2.2.26)	0x815D	0x0003 (**PtypInteger32**)	-1000
PidLidTaskFRecurring (section 2.2.2.2.28)	0x814B	0x000B (**PtypBoolean**)	0x00
PidLidCommonStart (section 2.2.2.1.3)	0x81BD	0x0040 (**PtypTime**)	<not found>
PidLidCommonEnd (section 2.2.2.1.4)	0x81BC	0x0040 (**PtypTime**)	<not found>
PidLidTaskGlobalId (section 2.2.2.2.32)	0x8211	0x0102 (**PtypBinary**)	0E B0 1E 03 85 02 EF 4B 9A 14 50 83 B3 BB 4D E9

The client will use the value of the **PidLidTaskGlobalId** property to locate the Task object locally and will then use the values of the other properties to copy to the local Task object.

The client uses a handle to the Tasks folder and the **RopGetContentsTable** ROP ([MS-OXCROPS] section 2.2.4.14) to get a handle to the **contents table** of the folder. Using this handle, the client uses the **RopSetColumns** ROP ([MS-OXCROPS] section 2.2.5.1).

Property	Property ID	Type
PidTagFolderId ([MS-OXCFOLD] section 2.2.2.2.1.5)	0x6748	0x0014 (**PtypInteger64**)
PidTagMid ([MS-OXCFXICS] section 2.2.1.2.1)	0x674A	0x0014 (**PtypInteger64**)

With the proper column set, the client can now search for the local Task object whose **PidLidTaskGlobalId** property matches the one found in the embedded Task object. The client performs the search with the **RopFindRow** ROP ([MS-OXCROPS] section 2.2.5.13).

Condition type	Relational operator	Property ID	Property data
0x04 (RES_PROPERTY)	0x04 (RELOP_EQ)	0x8211 (**PidLidTaskGlobalId**)	0E B0 1E 03 85 02 EF 4B 9A 14 50 83 B3 BB 4D E9

Having completed the search, the client releases the handle to the contents table by using the **RopRelease** ROP ([MS-OXCROPS] section 2.2.15.3).

If the search succeeded, the client will have located the **PidTagFolderId** and **PidTagMid** properties for the local Task object. The client uses these values to open a handle to the local Task object by using the **RopOpenMessage** ROP ([MS-OXCROPS] section 2.2.6.1). The client will now use the **RopSetProperties** ROP ([MS-OXCROPS] section 2.2.8.6) to update the properties of the local Task object, copying the properties from the embedded Task object, as appropriate.

Property	Property ID	Type	Value
PidLidTaskStatus	0x8146	0x0003 (**PtypInteger32**)	0 (Not started)
PidLidPercentComplete	0x8147	0x0005 (**PtypFloating64**)	0.0 (0%)
PidLidTaskActualEffort	0x814D	0x0003 (**PtypInteger32**)	0
PidLidTaskEstimatedEffort	0x814E	0x0003 (**PtypInteger32**)	0
PidLidTaskAccepted	0x82C2	0x000B (**PtypBoolean**)	0x01
PidLidTaskMultipleRecipients	0x814F	0x0003 (**PtypInteger32**)	0
PidLidTaskUpdates	0x82C3	0x000B (**PtypBoolean**)	0x01
PidLidTaskStatusOnComplete	0x82C4	0x000B (**PtypBoolean**)	0x01
PidLidTaskComplete	0x8149	0x000B (**PtypBoolean**)	0x00
PidLidTaskFFixOffline	0x8156	0x000B (**PtypBoolean**)	0x00
PidLidTaskOwnership	0x8154	0x0003 (**PtypInteger32**)	1 (task assigner's copy)
PidLidTaskAcceptanceState	0x8151	0x0003 (**PtypInteger32**)	2 (Accepted)
PidLidTaskHistory	0x8150	0x0003	1 (Accepted)

Property	Property ID	Type	Value
		(**PtypInteger32**)	
PidLidTaskLastUpdate	0x8153	0x0040 (**PtypTime**)	2008/02/19
PidLidTaskLastUser	0x8152	0x001F (**PtypString**)	"Scott Bishop"
PidLidTaskLastDelegate	0x82C5	0x001F (**PtypString**)	"Scott Bishop"
PidLidTaskRole	0x8157	0x001F (**PtypString**)	""
PidLidTaskVersion	0x8158	0x0003 (**PtypInteger32**)	4
PidLidTaskState	0x8148	0x0003 (**PtypInteger32**)	3 (task assigner's copy of an accepted Task object)
PidLidTaskAssigner	0x8159	0x001F (**PtypString**)	""
PidLidTaskOwner	0x801B	0x001F (**PtypString**)	"Scott Bishop"
PidLidTaskFCreator	0x82CA	0x000B (**PtypBoolean**)	0x01
PidLidTaskOrdinal	0x815D	0x0003 (**PtypInteger32**)	-1000
PidLidTaskFRecurring	0x814B	0x000B (**PtypBoolean**)	0x00
PidLidTaskGlobalId	0x8211	0x0102 (**PtypBinary**)	0E B0 1E 03 85 02 EF 4B 9A 14 50 83 B3 BB 4D E9

The client saves and closes the local Task object, embedded Task object, and Attachment object by using, in order, the following operations: **RopSaveChangesMessage** ([MS-OXCROPS] section 2.2.6.3) with the handle to the local Task object, **RopRelease** with the handle to the embedded Task object, **RopRelease** with the handle to the Attachment object, and **RopRelease** with the handle to the local Task object.

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to the Task-Related Objects Protocol. General security considerations pertaining to the underlying transport apply, as described in [MS-OXCMSG].

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

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

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

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

<1> Section 2.2.2.2.23: In Office Outlook 2003, Office Outlook 2007, Outlook 2010, and Outlook 2013 the distinction of Task object as a team task has no meaning.

<2> Section 3.1.4.3: Office Outlook 2003, Office Outlook 2007, Outlook 2010, and Outlook 2013 set the rendering position and hidden flag in a separate **RopSetProperties** **ROP request** ([MS-OXCROPS] section 2.2.8.6) after opening the embedded message.

<3> Section 3.1.4.5: Office Outlook 2007, Outlook 2010, and Outlook 2013 do not create the local Task object when it cannot locate the local Task object.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

S

T

V