[MS-OXODOC]: Document Object Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- **Copyrights.** This protocol 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 protocols, and may distribute portions of it in your implementations of the protocols 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 protocol documentation.
- **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 protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, the protocols may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp). If you would prefer a written license, or if the protocols are not covered by the OSP, patent licenses are available by contacting protocol@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

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

Tools. This protocol documentation is 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. A protocol specification does 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.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability.
Microsoft Corporation	June 27, 2008	1.0	Initial Release.
Microsoft Corporation	August 6, 2008	1.01	Revised and edited technical content.
Microsoft Corporation	September 3, 2008	1.02	Updated references.
Microsoft Corporation	December 3, 2008	1.03	Updated IP notice.

Table of Contents

1 Introduction

This document specifies the Document Object protocol, which is an extension to the Message and Attachment Object protocol. The Document Object protocol defines **properties** that allow a messaging client to store and display ordinary files. For more information about the Message and Attachment Object protocol, see [MS-OXCMSG].

1.1 Glossary

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

attachment
Attachment object
folder
handle
message
message class
Message object
property

1.2 References

1.2.1 Normative References

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

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

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

[MS-OXOMSG] Microsoft Corporation, "E-Mail Object Protocol Specification", June 2008.

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

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

1.2.2 Informative References

None.

1.3 Protocol Overview

The Document Object protocol extends the Message and Attachment Object protocol by defining the Document object, which is a **Message object** with additional **properties**. For more information about the Message and Attachment Object protocol, see [MS-OXCMSG]. For more information about Message objects, see [MS-OXOMSG].

The Document object is used to represent a file, such as a document generated by a word-processing application. The file is embedded within the Document object; the embedded file is referred to as an **attachment**. Specifically, the Document object is a Message object that:

- Contains one file.
- Includes additional properties to describe the file.

A file represented by a Document object can be stored in a mail **folder** and retrieved from the mail folder. For example, a user might choose to store a few files in his or her mail folders so that the files can be accessed not just on one computer, but on any computer that provides access to the user's e-mail.

1.4 Relationship to Other Protocols

The Document Object protocol relies on the same protocols as the Message and Attachment Object protocol, which the Document Object protocol extends. For more information about the Message and Attachment Object protocol, see [MS-OXCMSG].

1.5 Prerequisites/Preconditions

The Document Object protocol assumes that the messaging client has a mail **folder** open. Before sending requests to the server, the client MUST obtain a **handle** to the **Message object** that is used in **property** operations.

1.6 Applicability Statement

The client can utilize this protocol to store ordinary files in a user's mail **folders** and to expose the files that are stored in the mail folders.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

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

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Document Object protocol uses the same underlying transport as the Message and Attachment Object Protocol, as specified in [MS-OXCMSG].

2.2 Message Syntax

2.2.1 Inherited Properties

Unless otherwise specified below, a Document object MUST adhere to all **property** constraints that are specified in [MS-OXPROPS] and [MS-OXCMSG]. A Document object MAY <1> <2 >also contain other properties defined in [MS-OXPROPS], but these properties have no impact on the Document Object protocol.

2.2.2 Document Object Properties and Format

The **properties** specified in the following sections are specific to a Document object. However, additional properties MAY be set on the **message** by a messaging client.

2.2.2.1 PidTagMessageClass Property

A **PtypString** value that specifies the type of the **Message object**. For a message to be treated like a Document object by a messaging application, the value of this **property** MUST begin with "IPM.Document.". The value of the sub-string that follows "IPM.Document" depends on the type of the attached file. For more details about the **PidTagMessageClass** property, see [MS-OXCMSG] section 2.2.1.3.

2.2.2.2 PidTagDisplayName Property

A **PtypString** value that specifies the name of the **attachment**. A Document object SHOULD have the **PidTagDisplayName** property defined. This SHOULD be the name of the file that is attached.

2.2.2.3 Attachment to the Message Object

A Document object SHOULD have only one **attachment** and MUST have at least one attachment. For more details about how attachments are stored within a **message**, see [MS-OXCMSG]. If there is more than one attachment, then a messaging client can choose to display an error or to pick any attachment in the collection to use as the main file. However, a Document object only makes sense if there is only one attachment. If there are zero attachments, then the messaging client SHOULD cause an error when the item is invoked.

2.2.2.4 Document-Specific Properties

A Document object encapsulates the behavior of the attached file. As such, many **properties** on a file SHOULD be promoted as properties on the **message** itself if those properties exist on

the file. The following is a list of such properties that SHOULD be set on the message if they exist on the attached file. For more details about these properties, see [MS-OXPROPS].

2.2.2.4.1 PidNameTitle

This is a string value that specifies the title of the file attached to the Document object. This value MAY be present.

2.2.2.4.2 PidNameSubject

This is a string value that specifies the subject of the file attached to the Document object. This value MAY be present.

2.2.2.4.3 PidNameAuthor

This is a string value that represents the author of the file attached to the Document object. This value MAY be present.

2.2.2.4.4 PidNameKeywords

This is a multi-string value that specifies the categories of the file attached to the Document object. This value MAY be present.

2.2.2.4.5 PidNameComments

This is a string value that specifies the comments of the file attached to the Document object. This value MAY be present.

2.2.2.4.6 PidNameTemplate

This is a string value that specifies the template of the file attached to the Document object. This value MAY be present.

2.2.2.4.7 PidNameLastAuthor

This is a string value that specifies the most recent author of the file attached to the Document object. This value MAY be present.

2.2.2.4.8 PidNameRevisionNumber

This is a string value that specifies the revision number of the file attached to the Document object. This value MAY be present.

2.2.2.4.9 PidNameApplicationName

This is a string value that specifies the application that might open the file attached to the Document object. This value MAY be present.

2.2.2.4.10 PidNameEditTime

This is a string value that specifies the time that the file was last edited. This value MAY be present.

2.2.2.4.11 PidNameLastPrinted

This is a **PtypTime** value that specifies the time that the file was last printed. This value MAY be present.

2.2.2.4.12 PidNameCreateDateTimeReadOnly

This is a **PtypTime** value that specifies the time that the file was first created. This value MAY be present.

2.2.2.4.13 PidNameLastSaveDateTime

This is a **PtypTime** property that specifies the time that the file was last saved. This value MAY be present.

2.2.2.4.14 PidNamePageCount

This is an Int32 value that specifies the page count of the file attached to the Document object. This value MAY be present.

2.2.2.4.15 PidNameWordCount

This is an Int32 value that specifies the word count of the file attached to the Document object. This value MAY be present.

2.2.2.4.16 PidNameCharacterCount

This is an Int32 value that specifies the character count of the file attached to the Document object. This value MAY be present.

2.2.2.4.17 PidNameSecurity

This is an Int32 value that specifies the security level of the file attached to the document object. This value MAY be present.

2.2.2.4.18 PidNameCategory

This is a string value that specifies the category of the file attached to the Document object. This value MAY be present.

2.2.2.4.19 PidNamePresentationFormat

This is a string value that specifies the presentation format of the file attached to the document object. This value MAY be present.

2.2.2.4.20 PidNameManager

This is a string value that specifies the manager of the file attached to the document object. This value MAY be present.

2.2.2.4.21 PidNameCompany

This is a string value that specifies the company for which the file was created. This value MAY be present.

2.2.2.4.22 PidNameByteCount

This is an Int32 value that specifies the size, in bytes, of the file attached to the Document object. This value MAY be present.

2.2.2.4.23 PidNameLineCount

This is an Int32 value that specifies the number of lines in the file attached to the Document object. This value MAY be present.

2.2.2.4.24 PidNameParagraphCount

This is an Int32 value that specifies the number of paragraphs in the file attached to the Document object. This value MAY be present.

2.2.2.4.25 PidNameSlideCount

This is an Int32 value that specifies the number of slides in the file attached to the Document object. This value MAY be present.

2.2.2.4.26 PidNameNoteCount

This is an Int32 value that specifies the number of notes in the file attached to the Document object. This value MAY be present.

2.2.2.4.27 PidNameHiddenCount

This is an Int32 value that specifies the hidden value of the file attached to the Document object. This value MAY be present.

2.2.2.4.28 PidNameMultimediaClipCount

This is an Int32 value that specifies the number of multimedia clips in the file attached to the Document object. This value MAY be present.

3 Protocol Details

3.1 Document Object Client Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

3.1.1.1 Managing Document Objects

A messaging client user can choose to create a Document object either programmatically or as a result of user interaction. Choosing either of these routes will result in the creation of a **message** that has certain **properties** that make the message a Document object. For instance, a user could drag a file from his or her desktop into a **folder** in the messaging client. The end result of this user interaction MAY be a Document object in that folder. Howa user interacts with this Document object is entirely up to the messaging client. One behavior can be as follows: When a user invokes (double-clicks) this message, the file is directly opened instead of first opening the message and then necessitating another click on the **attachment**.

3.1.2 Timers

None.

3.1.3 Initialization

A Document object is initialized or created either programmatically or as a result of user interaction. In either case, a Document object is created with the correct **attachment** and properties being set on the message. For more details about the properties, see section 2.2.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Creating a Document Object

One of the ways that a Document object can be created by a messaging client is by dragging any file from the user's desktop (or any file **folder**) into a mail folder in the messaging client. This action SHOULD cause a Document object to be created with one **attachment** and the **PidTagMessageClass** property and the **PidTagDisplayName** property correctly set. <3> <4> For more details about **properties**, see section 2.2. A Document object can also be created programmatically.

3.1.4.2 Invoking a Document Object

In a messaging client, when a **message** is invoked, the messaging client first needs to get the message type. This involves getting the **PidTagMessageClass** property. <5> In the case of Document objects, the value of **PidTagMessageClass** MUST begin with "IPM.Document". If the value does not begin with "IPM.Document", then the message is not a Document object, and the messaging client will handle it in a different way. If the value of **PidTagMessageClass** does begin with "IPM.Document", then the message is a Document object, and the messaging client will proceed to retrieve the **attachment** from the message's attachment collection and open that attachment. <6> <7> There SHOULD be only one attachment. For more details about attachments, see [MS-OXCMSG].

3.1.5 Message Processing Events and Sequencing Rules

None.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

4.1 Example PidTagMessageClass Values for Different File Types

The following table shows example **message class** values for different file types.

File extension	PidTagMessageClass value
.doc	IPM.Document.Word.Document.8
.docx	IPM.Document.Word.Document.12
.xls	IPM.Document.Excel.Sheet.8
.xlsx	IPM.Document.Excel.Sheet.12
.ppt	IPM.Document.PowerPoint.Show.8
.pptx	IPM.Document.PowerPoint.Show.12
.txt	IPM.Document.txtfile

4.2 Example for Creating a New Document Object Item

Joe drags a file (for example, testDocObj.txt) from his desktop into one of his mail **folders**. The following sub-sections provide descriptions of what a client might do to accomplish Joe's intentions, and the responses that a server might return.

4.2.1 Creating the Object

To create a Document object, the protocol client uses the **RopCreateMessage** operation. The protocol server returns a success code and a **handle** to a **Message object**.

4.2.2 Attachment Details

The client uses the **RopCreateAttachment** operation to create the **Attachment object**. Then, the protocol client uses the **RopOpenStream** and **RopSetStreamSize** operations followed by **RopWriteStream** to write out the contents of the file into the **attachment**.
The client then uses **RopSetProperties** to set various **properties** on the attachment. The following table shows just some of the properties that would be set on the attachment.

Property	Property ID	Type	Value
PidTagAttachLongFilename	0x3707	0x001f (string)	"testDocObj.txt"
PidTagAttachExtension	0x3703	0x001f (string)	".txt"
PidTagCreationTime	0x3007	0x0040(date	2008/02/15 19:57:52.557

		and time)	

Now the protocol client uses **RopSaveChangesAttachment** to save the attachment.

4.2.3 Setting Properties on the Document Object

The protocol client uses the **RopSetProperties** operation to transmit his data to the protocol server. The following table shows some of the relevant **properties** that need to be set for a Document object.

Property	Property ID	Type	Value
PidTagDisplayName	0x3001	0x001f (PT_UNICODE)	"testDocObj.txt"
PidTagMessageClass	0x001a	0x001f	"IPM.Document.txtfile"

4.2.4 Final Save

The protocol client uses the **RopSaveChangesMessage** operation to commit the **properties** on the protocol server and then uses **RopRelease** to release the object. The values of some properties will change during the execution of **RopSaveChangesMessage**, but none of the properties specified in this protocol will change.

5 Security

5.1 Security Considerations for Implementers

Document objects store files as **attachments**. These files can be any files on the hard drive. When a user invokes a Document object, one behavior is to open up the attached file directly. There is a security implication here in that this file could do harmful things when invoked. While this is less of an issue for a user's personal mail **folders**, it becomes much more of an issue for public mail folders. It is up to the messaging client to choose what kind of behavior to follow when a user clicks on one of these Document objects.

5.2 Index of Security Parameters

Security Parameter	Section
PidNameSecurity property	2.2.2.4.17

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> Section 2.2.1: Outlook 2003 SP3 and Outlook 2007 SP1 sometimes set the following **properties** regardless of user input; their values have no meaning in the context of this protocol:
PidLidAgingDontAgeMe, **PidLidCurrentVersion**, **PidLidCurrentVersionName**, **PidLidPrivate**, **PidLidSideEffects**, **PidTagAlternateRecipientAllowed**, **PidTagClientSubmitTime**, **PidTagDeleteAfterSubmit**, **PidTagImportance**, **PidTagMessageDeliveryTime**, **PidTagPriority**, **PidTagReadReceiptRequested**, **PidTagSensitivity**, **PidLidReminderDelta**, **PidLidReminderSet**, **PidLidReminderNextTime**, and **PidLidTaskMode**.

<2> Section 2.2.1: Outlook 2007 SP1 sets the following **properties** regardless of user input; their values have no meaning in the context of this protocol:
PidLidPercentComplete, **PidLidTaskActualEffort**, **PidLidTaskComplete**, **PidLidTaskAssigner**, **PidLidTaskAcceptanceState**, **PidLidTaskEstimatedEffort**, **PidLidTaskFFixOffline**, **PidLidTaskFRecurring**, **PidLidTaskNoCompute**, **PidLidTaskOrdinal**, **PidLidTaskOwnership**, **PidLidTaskRole**, **PidLidTaskState**, **PidLidTaskStatus**, **PidLidTaskVersion, PidLidTeamTask**, and **PidLidValidFlagStringProof.**

<3> Section 3.1.4.1: Outlook uses the Windows registry to determine the value of the file-type part of the **PidTagMessageClass** property.

<4> Section 3.1.4.1: When a file is dragged into a mail folder in Outlook, a Document object is created with a **PidTagMessageClass** value of "IPM.Document.<fileType>". Outlook determines the "<fileType>" sub-string by doing the following:

1. Determine the file extension of the attached file.
2. Using the file extension, reference the registry under HKEY_CLASSES_ROOT\<fileExtension>. For example, if the attached file is a text file, reference HKEY_CLASSES_ROOT\.txt.
3. Look at the (default) registry value under HKEY_CLASSES_ROOT\<fileExtension>. and use the value as the file-type part of the **PidTagMessageClass** property. For

example, the value under HKEY_CLASSES_ROOT\.txt is "txtfile", so the value of PidTagMessageClass would be "IPM.Document.txtfile".

4. If the file extension is not found in the registry, Outlook simply uses the actual file extension as the file-type part of the **PidTagMessageClass** property. For instance, if the file extension is .zzz, and .zzz is not found in the registry, Outlook would create a **PidTagMessageClass** value that is "IPM.Document.*.zzz". Notice the "*.zzz" after "IPM.Document.".

This is just an implementation followed by Outlook. Non-Windows **messaging clients** that might not want to reference the registry could use a more prescriptive approach, where a predefined list of file extensions can be used to generate the corresponding file-type parts of PidTagMessageClass.

<5> Section 3.1.4.2: Outlook uses the Windows registry to interpret the value of the file-type part of the **PidTagMessageClass** property.

<6> Section 3.1.4.2: When a Document object in a user's mail **folder** is invoked, Outlook will open the underlying **attachment** to the **message** directly, thus making the feature behave in the most optimal fashion from a user's perspective.

<7> Section 3.1.4.2: The default behavior in Outlook Web Access is to treat the Document object as an e-mail **message**. When the user invokes it, the **attachment** is not opened. Instead, the message is opened, and the user can then choose to invoke the attachment in the message.

Index