[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. 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/default.mspx). 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.

Preliminary Documentation. This documentation is preliminary documentation for these protocols. Since the documentation may change between this preliminary version and the final version, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Tools. This protocol documentation is intended for use in conjunction with publicly available standard specifications and networking programming art, and assumes that the reader is either 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 a Licensee to develop an implementation. Licensees who have access to Microsoft programming tools and environments are free to take advantage of them.

Revision Summary			
Author	Date	Version	Comments
Microsoft Corporation	April 4, 2008	0.1	Initial Availability

Table of Contents

1 Introduction

The Document Object protocol is an extension to the Message Object protocol. The Document Object protocol supports the use of a Message Object to represent a document, such as a file generated by a word-processing application. The Document Object protocol defines a set of properties that can be used to describe the document embedded within the Document Object.

1.1 Glossary

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

bit
byte

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

attachment
folder
handle
message
message class
message object
named 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", April 2008.

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

[MS-OXOMSG] Microsoft Corporation, "E-mail Object Protocol Specification", April 2008.

[MS-OXPROPS] Microsoft Corporation, "Office Exchange Protocols Master Property List Specification", April 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 (Synopsis)

The Document Object protocol allows users to store files in mail folders. For example, a user might choose to store a few files in his or her mail folder so that they can be accessed not just on one computer, but on any computer that has access to his or her e-mail.

The Document Object **message** is a special type of message with certain properties and a data set that allows a messaging client to display it as a file.

1.4 Relationship to Other Protocols

The Document Object protocol specification relies on the following:

- An understanding of the **message object** (see [MS-OXOMSG] and [MS-OXCMSG])
- An understanding of **attachments** (see [MS-OXCMSG])
- An understanding of message types. Document Object messages are a type of message that has a special value for the PidTagMessageClass property.

1.5 Prerequisites/Preconditions

The Document Object protocol specification assumes the messaging client has a mail folder open.

1.6 Applicability Statement

The client can utilize this protocol to expose ordinary files (from a computer) in your 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 protocols specified in [MS-OXCMSG].

2.2 Message Syntax

Before sending requests to the server, the client MUST obtain a **handle** to the message object used in property operations.

2.2.1 Inherited Properties

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

2.2.2 Document Object Properties and Format

The properties specified below are specific to a Document Object message. However, additional properties MAY be set on the message by a messaging client.

2.2.2.1 PidTagMessageClass Property

For a message to be treated like a Document Object message by a messaging application, the value of this property MUST begin with "IPM.Document.". The rest of the string following "IPM.Document" is determined by the type of the attached file. For more information, see section [*PidTagMessageClass*] for examples.

2.2.2.2 PidTagDisplayName Property

A Document Object message SHOULD have the PidTagDisplayName property defined. This SHOULD be the file name of the file that is attached.

2.2.2.3 Attachment to the Message Object

A Document Object message SHOULD have only one attachment and MUST have at least one attachment. To understand 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 could choose to pick any attachment in the collection to use as "the main file". However, a Document Object message only makes sense if there is only one attachment, which why it SHOULD have only one attachment. If there are zero attachments, the messaging client SHOULD cause an error when the item is invoked.

2.2.2.4 Document Specific Properties

A Document Object message encapsulates the behavior of the attached document (or file). As such, many properties on a document or 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 document, which are further documented in [MS-OXPROPS].

2.2.2.4.1 PidNameTitle

This is a string value that represents the title property 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 represents the subject property 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 property 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 represents the Categories property 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 represents the comments property 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 represents the template property 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 represents the Last Author property of the file attached to the document object. This property represents the name of the person that last authored the file. This value MAY be present.

2.2.2.4.8 PidNameRevisionNumber

This is a string value that represents the revision number property 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 represents the Application Name property of the file attached to the document object. This property represents the name of the application that might open the file. This value MAY be present.

2.2.2.4.10 PidNameEditTime

This is a string value that represents the time that the document was last edited. This value MAY be present.

2.2.2.4.11 PidNameLastPrinted

This property represents the value when the file was last printed. This value MAY be present.

2.2.2.4.12 PidNameCreateDateTimeReadOnly

This property represents the value when the file was first created. This value MAY be present.

2.2.2.4.13 PidNameLastSaveDateTime

This property represents the value when the file was last saved. This value MAY be present.

2.2.2.4.14 PidNamePageCount

This is an Int32 value that represents 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 represents 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 represents 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 represents the security property 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 represents 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 represents 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 represents the manager property 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 represents the company property of the file attached to the document object. This value MAY be present.

2.2.2.4.22 PidNameByteCount

This is an Int32 value that represents the byte count 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 represents the line count of the file attached to the document object. This value MAY be present.

2.2.2.4.24 PidNameParagraphCount

This is an Int32 value that represents the paragraph count of the file attached to the document object. This value MAY be present.

2.2.2.4.25 PidNameSlideCount

This is an Int32 value that represents the slide count of the file attached to the document object. This value MAY be present.

2.2.2.4.26 PidNameNoteCount

This is an Int32 value that represents the note count of the file attached to the document object. This value MAY be present.

2.2.2.4.27 PidNameHiddenCount

This is an Int32 value that represents the Hidden property 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 represents the multimedia clip count of the file attached to the document object. This value MAY be present.

3 Protocol Details

3.1 Document Object Message 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 Object Messages

A messaging client user can choose to create a Document Object message either programmatically or as a result of user interaction. As specified in section 2.2, choosing either of these routes will result in a message being created that has certain properties that make the message a Document Object message. For instance, a user could drag a file from his desktop into a folder in the messaging client. The end result of this user interaction MAY be a Document Object message in that folder. Now, how does a user interact with this Document Object message? That behavior is a design that is entirely up to the messaging client. One behavior can be that 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 message is created with the correct attachment and properties being set on the message. See section 2.2 for more details.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Creating a Document Object Message

One of the ways that a Document Object message 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 message to be created with one attachment and the PidTagMessageClass and PidTagDisplayName property correctly set (see section 2.2). As another example, a Document Object message can be created programmatically.

3.1.4.2 Invoking a Document Object Message

So, what happens when a Document Object message is invoked? 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 ([MS-OXOMSG] and [MS-OXPROPS]). In the case of Document Object messages, this value MUST begin with "IPM.Document". If this value does not begin with "IPM.Document" then it is not a Document Object message, and the messaging client will handle it in a different way. If the PidTagMessageClass does begin with "IPM.Document", then it is a Document Object message, and the messaging client will proceed to retrieve the attachment (there SHOULD be only one) in the message's attachment collection ([MS-OXCMSG]) and open that attachment.

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 is a description of what a protocol client might do to accomplish Joe's intentions and the responses a protocol server might return.

4.2.1 Creating the Object

To create a document object, the protocol client uses RopCreateMessage. The protocol server returns a success code and a handle to a message object.

4.2.2 Attachment Details

Now the protocol client uses RopCreateAttachment to create the attachment object. Then the protocol client uses RopOpenStream and RopSetStreamSize followed by RopWriteStream to write out the contents of the file into the attachment.

Now the protocol client 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 and time)	2008/02/15 19:57:52.557

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

4.2.3 Setting Properties on the Document Object

Now the protocol client uses RopSetProperties 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	"testDocObj.txt"

		(PT_UNICODE)	
PidTagMessageClass	0x001a	0x001f	“IPM.Document.txtfile”

4.2.4 Final Save

The protocol client uses RopSaveChangesMessage 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 Object messages store files as attachments. These files can be any files on the hard drive. When a user invokes a Document Object message, 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 Object messages.

5.2 Index of Security Parameters

Security Parameter	Section
Security 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 (SP3) applied
- Exchange 2003 with Service Pack 2 (SP2) applied
- Office 2007 with Service Pack 1 (SP1) applied
- Exchange 2007 with Service Pack 1 (SP1) 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.

6.1 Microsoft Office Outlook Behavior

When a Document Object message in a user's mail folder is invoked, Microsoft Office 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.

6.2 Microsoft Outlook Web Access (Microsoft Exchange)

The default behavior for a Document Object message in Outlook Web Access is to treat the Document Object message 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.

6.3 PidTagMessageClass Values in Microsoft Outlook

Outlook uses the Windows registry to determine and interpret the value of the "file-type" part of the PidTagMessageClass property.

6.3.1 Setting PidTagMessageClass When Creating a Document Object Message

If a text file (with extension .txt) is dragged into a mail folder in Outlook, then the Document Object message that will be created will have a PidTagMessageClass value of "IPM.Document.txtfile". Outlook produces this value by doing the following:

1. Determine the file extension of the file (for example, .txt)
2. Reference the registry under HKEY_CLASSES_ROOT\<fileExtension>. In this case, reference HKEY_CLASSES_ROOT\.txt
3. Look at the value of the (default) registry value under HKEY_CLASSES_ROOT\<fileExtension>. The value in this example is "txtfile" and is used as the file extension part of PidTagMessageClass. If the file extension is not found in the registry, Outlook simply uses the actual file extension as part of the PidTagMessageClass value. For instance, if the file extension was ".zzz", and ".zzz" was not found in the registry, Outlook would create a PidTagMessageClass value that is "IPM.Document.*.zzz". Notice the "*.zzz" after "IPM.Document." .
4. So in this example, the PidTagMessageClass would be "IPM.Document.txtfile"

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 generate corresponding PidTagMessageClass values.

6.3.2 Getting the Icon to Display from PidTagMessageClass

Outlook uses the "file-type" part of the PidTagMessageClass property to determine the icon to display. Outlook uses this mechanism to display the icon of the attached file. To determine the icon, Outlook does the following:

1. Retrieves the PidTagMessageClass property of the Document Object message
2. Determines the value after "IPM.Document.". For example, if the attached file were a text file (with .txt as the extension), then the PidTagMessageClass value would be "IPM.Document.txtfile", and the file-type part would be "txtfile".

3. Uses the registry to access HKEY_CLASSES_ROOT\<filetype>\DefaultIcon to retrieve the icon for this file type. In the above example, the registry needs to look at HKEY_CLASSES_ROOT\txtfile\DefaultIcon. If the registry key is not found, then a default icon (as chosen by the messaging client) could be used.
4. Accesses the (default) value for this registry key. It will give you the location of the icon to display. For example, the value will look something like this: "%SystemRoot%\system32\imageres.dll,-102". That means that the icon to display is a resource in the imageres.dll file with resource ID 102.

6.4 Other Possible Ways to Display the Icon of a Document Object Message

A messaging client can display the Document Object message in a mail folder as if they were files on the hard drive. As such, they can choose to display a relevant icon for the attached file of the Document Object message.

6.4.1 Directly Access the File Name of the Attachment

One way to directly access the file name of the attachment might be to simply access the attached file name, and use the extension to determine the icon to display.

<1> "Microsoft Office Outlook 2003" and "Microsoft Office Outlook 2007" 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, PidLidTaskMode

<2> "Microsoft Office Outlook 2007" 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, PidLidValidFlagStringProof

Index