

# [MS-OXOUM]: Voice Mail and Fax Objects 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.msp>). 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](mailto: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

Preliminary

## Table of Contents

<b>1</b>	<b><i>Introduction</i></b> .....	<b>4</b>
1.1	Glossary.....	4
1.2	References.....	5
1.2.1	Normative References.....	5
1.2.2	Informative References.....	6
1.3	Protocol Overview (Synopsis).....	6
1.4	Relationship to Other Protocols.....	6
1.5	Prerequisites/Preconditions.....	6
1.6	Applicability Statement.....	6
1.7	Versioning and Capability Negotiation.....	6
1.8	Vendor-Extensible Fields.....	6
1.9	Standards Assignments.....	6
<b>2</b>	<b><i>Messages</i></b> .....	<b>7</b>
2.1	Transport.....	7
2.1.1	Inherited Properties.....	7
2.1.2	Message-Specific Properties.....	7
2.2	UI Configuration.....	9
<b>3</b>	<b><i>Protocol Details</i></b> .....	<b>10</b>
3.1	Client Details.....	10
3.1.1	Abstract Data Model.....	11
3.1.2	Timers.....	11
3.1.3	Initialization.....	11
3.1.4	Higher-Layer Triggered Events.....	11
3.1.5	Message Processing Events and Sequencing Rules.....	11
3.1.6	Timer Events.....	11
3.1.7	Other Local Events.....	11
3.2	Server Details.....	11
<b>4</b>	<b><i>Protocol Examples</i></b> .....	<b>11</b>
4.1	Playing a Voice Message.....	11
4.1.1	Down-Level Experience.....	11
4.1.2	Up-Level Experience.....	12
<b>5</b>	<b><i>Security</i></b> .....	<b>12</b>
5.1	Security Considerations for Implementers.....	12
5.2	Index of Security Parameters.....	12
<b>6</b>	<b><i>Appendix A: Office/Exchange Behavior</i></b> .....	<b>12</b>
<b>7</b>	<b><i>Index</i></b> .....	<b>14</b>

# 1 Introduction

## 1.1 Glossary

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

**attachment**  
**message**  
**message object**  
**property**  
**property tag**  
**recipient**  
**special folder**

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

**Unicode**  
**ANSI**  
**BYTE**  
**DWORD**  
**ULONG**  
**Boolean**

The following terms are specific to this document:

**audio notes:** Textual notes that can be attached to a voice message.

**fax message:** A message stored in the server for a user that contains image content received from a calling fax machine; a digital representation of a typical fax message.

**G.711:** An ITU-T standard for audio compression typically used in telephony systems. There are two different implementations: a-law and m-law.

**GSM 6.10:** A form of audio compression used by most European wireless telephone systems.

**missed call notification:** A message stored in the server for a user intended to convey information about a call that was missed. The message contains information about the calling party, if available, and the time of call, but does not contain audio content because the calling party chose not to record a message.

**voice message:** A message stored in the server for a user that contains audio content recorded by a calling party.

**WAV:** The main format for storing audio on Windows computers.

**WMA 9 Voice:** A form of audio compression released by Microsoft as part of the Windows Media Audio 9 SDK tool kit.

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

## 1.2 References

### 1.2.1 Normative References

[G711] ITU-T, "Pulse Code Modulation (PCM) of Voice Frequencies", Recommendation G.711, June 1990, <http://www.itu.int/rec/T-REC-G.711/e>.

[GSM610] ETSI, "European digital cellular telecommunications system (Phase 1); Full rate speech; Transcoding (GSM 06.10)", February 1992, [http://pda.etsi.org/pda/home.asp?wki\\_id=v9jLO9Nb7wSVbbYKNyW](http://pda.etsi.org/pda/home.asp?wki_id=v9jLO9Nb7wSVbbYKNyW).

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

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

[MS-OXGLOS] Microsoft Corporation, "Office Exchange Protocols Master Glossary", April 2008.

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

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

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

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

[MS-OXWUMS] Microsoft Corporation, "Voice Mail Settings Web Service Protocol Specification", April 2008.

[MSDN-ASF] Microsoft Corporation, "Advanced Systems Format (ASF) Specification", <http://go.microsoft.com/fwlink/?LinkId=114334>.

[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>.

[WAVE] IBM Corporation and Microsoft Corporation, "Multimedia Programming Interface and Data Specifications 1.0", August 1991, <http://www-mmmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/Docs/riffmci.pdf>.

[MS-OXOUM] - v0.1

Voice Mail and Fax Objects Protocol Specification  
Copyright © 2008 Microsoft Corporation.  
Release: Friday, April 4, 2008

### 1.2.2 Informative References

[MSDN-WMSSDK] Microsoft Corporation, "Windows Media Services 9.0 Series SDK", <http://go.microsoft.com/fwlink/?LinkId=114332>.

### 1.3 Protocol Overview (Synopsis)

Unified Messaging objects are items created on behalf of telephone callers or fax senders by the server and stored in the called party's mailbox on the server.

There are three types of these objects created by the server: voice messages, fax messages, and missed call notifications as defined in section 1.1.

The Voice Mail and Fax Objects protocol specifies the properties on the aforementioned Unified Messaging items above and beyond common message properties defined in [MS-OXOMSG].

### 1.4 Relationship to Other Protocols

The Voice Mail and Fax Objects protocol specification relies on an understanding of how to work with properties, folders, messages, and attachments (for more details see [MS-OXPROPS], [MS-OXOSFLD], and [MS-OXCMSG]).

The Voice Mail and Fax Objects protocol uses the Message and Attachment Object protocol as specified in [MS-OXCMSG] as a transport protocol between the client and the server.

### 1.5 Prerequisites/Preconditions

None.

### 1.6 Applicability Statement

This protocol can be used to model the electronic equivalent to telephony-based messages such as voice messages, fax messages, and missed call notifications.

### 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

There are three types of Unified Messaging objects: voice messages, missed call notifications, and fax messages.

### 2.1 Transport

The Voice Mail and Fax Objects protocol uses protocols specified in [MS-OXCMSG] to create and store the three types of Unified Messaging objects.

#### 2.1.1 Inherited Properties

None.

#### 2.1.2 Message-Specific Properties

Note that, unlike many other client-server objects, the server is actually the one creating the Unified Messaging objects. The server **MUST** include the common properties as specified in [MS-OXCMSG]. The server **SHOULD** also set the common properties as specified in [MS-OXOMSG].

##### 2.1.2.1 Message Classes

For voice messages, the server **MUST** set the value of the **PidTagMessageClass** property to the following:

- IPM.Note.Microsoft.Voicemail.UM.CA for original messages taken with audio content via the telephone.
- IPM.Note.Microsoft.Voicemail.UM for original messages taken with audio content via the telephone, but not as a result of call answering (i.e., the recipient's phone was not rung).
- The value of the original **PidTagMessageClass** property suffixed with .Microsoft.Voicemail for messages with audio content created as responses to other messages. For example, a voice reply to a message of type IPM.Note **MUST** have type IPM.Note.Microsoft.Voicemail.

For fax messages:

- IPM.Note.Microsoft.FAX.CA.

For missed call notifications:

- IPM.Note.Microsoft.Missed.Voice

##### 2.1.2.2 Attachments

For messages with audio content, the server **MUST** add the audio content as a file attachment on the message, in accordance with the procedures for attachment handling specified in [MS-OXCMSG].

The attachment file itself MUST be in either the WAV file format or in the ASF file format (see [WAV] and [ASF]).

- If in the WAV format, the audio codec MUST be either G.711 a-law or G.711 m-law or GSM 6.10 (see [G711] and [GSM610]).
- If in the ASF file format, the codec MUST be the WMA 9 Voice codec (see [WMSSDK]).

In addition to the common properties on the attachment, the server MUST set two properties on the attachment as follows:

- **PidTagAttachLongFileName**: Set to a unique name in the attachment collection of the message. The filename MUST be unique in order for the attachment order logic described in section 2.1.2.3 to function properly. The file extension MUST be “.wav” for files in the WAV format and MUST be “.wma” for files in the ASF format.
- **PidTagAttachMimeTag**: Set to reflect the audio content type of the message.
  - For WMA 9 Voice encoded messages, this MUST be “audio/wma”.
  - For GSM 6.10 encoded messages, this MUST be “audio/gsm”.
  - For G.711 encoded messages, this MUST be “audio/wav”.

In some situations, a client or server MAY add more than one audio attachment to a given message. For example, a voice reply to a voice message MAY include the original voice content for reference. In such situations, the server SHOULD add an attachment for each voice segment and follow the order specifying procedure detailed discussed in section 2.1.2.3.

### 2.1.2.3 Attachment Order

The server MUST create the **PidTagVoiceMessageAttachmentOrder** property as defined in [MS-OXPROPS] on any message containing audio attachments. Both clients and servers SHOULD consult this property to determine what order to play any audio attachments, including the trivial case of only one attachment.

The content of the property is a list of **PidTagAttachLongFileName** values for audio file attachments that are to be played as part of the message. The order MUST be the reverse order in which the attachments were added; that is, most recently added message first, next most recently added second, and so on.

The file names MUST be separated by semi-colons. Each filename MAY be prefixed or suffixed with whitespace. The first file name in the list MAY be preceded by a semi-colon and the last file name in the list MAY be suffixed with a semi-colon.

For example, for a message that contains only one voice file attachment, the **PidTagAttachLongFileName** for which is vm.wav, acceptable values for **PidTagVoiceMessageAttachmentOrder** are illustrated by, but not limited to, the following:

- vm.wav

- ;vm.wav
- ; vm.wav
- vm.wav;
- vm.wav ;

Or, for example, a message contains three attachments, the **PidTagAttachLongFileName** properties for which are “vm1.wav”, “vm2.wav”, and “vm3.wav”. The files were added in the order “vm1.wav”, then “vm2.wav”, then “vm3.wav”. Acceptable values for **PidTagVoiceMessageAttachmentOrder** are illustrated by, but not limited to, the following:

- vm3.wav;vm2.wav;vm1.wav
- vm3.wav; vm2.wav; vm1.wav
- ;vm3.wav;vm2.wav;vm1.wav
- Vm3.wav;vm2.wav;vm1.wav;
- ;vm3.wav;vm2.wav;vm1.wav;

#### 2.1.2.4 Audio Notes

The client MAY allow a user to annotate a voice message with textual information after it has been delivered to the user’s mailbox. For example, some users might find it useful to take note of a telephone number or name heard in the audio content of the message.

If the client saves these textual notes on the message, it MUST set the **PidNameAudioNotes** property, as described in [MS-OXPROPS], to the value of those textual notes.

## 2.2 UI Configuration

A client application MAY want to surface an enhanced user interface (UI) for Message objects with the message classes described in section 2.1.2.1 for some users and not for others; additionally, the client MAY want to show UI configuration information related to a user’s telephony experience for some users and not for others. The server SHOULD store settings for these options on a per user basis and the client MUST consult these settings if it wants to implement the aforementioned UI segmentation.

A scenario where this could be useful is if a certain group of users are not provisioned by their administrator to receive the message classes described in section 2.1.2.1 and/or are not provisioned to have telephony access to their messages.

If the client or server sets or uses this configuration information, it MUST treat it as a Dictionary Stream using the Configuration Information Protocol defined in [MS-OXOCFG].

The Dictionary Stream object MUST be stored in the Inbox special folder.

The Dictionary Stream MUST have the **PidTagMessageClass** string property set on it. The value of the property MUST be IPM.Configuration.UMOLK.UserOptions.

The Dictionary SHOULD include following setting. If the setting does not appear in the Dictionary, or the Dictionary itself does not exist, the following default value SHOULD be assumed:

- outlookFlags
  - Name: (string) "outlookFlags"
  - Value: (32-bit integer). The least significant bit MUST correspond to whether the client SHOULD surface special UI for message classes described in section 2.1.2.1, with 1 corresponding to “do surface” and 0 corresponding to “do not surface”. The second-least significant bit MUST correspond to whether the client SHOULD surface telephony configuration UI to the user, with 1 corresponding to “do surface” and 0 corresponding to “do not surface”. Thus, the value MUST take one of four values:
    - “0x00000000”: Neither surface special UI for the message classes described in section 2.1.2.1 nor for telephony configuration purposes.
    - “0x00000001”: Surface special UI for the message classes described in section 2.1.2.1, but not for telephony configuration purposes.
    - “0x00000002”: Do not surface special UI for the message classes described in section 2.1.2.1, but do show special UI for telephony configuration purposes.
    - “0x00000003”: Surface special UI for the message classes described in section 2.1.2.1 and also for telephony configuration purposes.
  - Default: (32-bit integer) “0x00000000”

## 3 Protocol Details

### 3.1 Client Details

The client role is to display Unified Messaging objects described in section 2.1.2.1. There are two possible levels of client experience.

A “down-level” experience would do nothing apart from the basic client role specified in [MS-OXCMSG] for message objects. An example of this experience is included in section 4.1.1.

Alternatively, the client MAY provide an “up-level” experience for displaying Unified Messaging objects, including the ability to edit audio notes (section 2.1.2.4) and/or providing a means to automatically play back the audio content of a message using the attachments (section 2.1.2.2) and the attachment order information (section 2.1.2.3). An example of this experience is included in section 4.1.2.

### **3.1.1 Abstract Data Model**

See [MS-OXOMSG].

### **3.1.2 Timers**

None.

### **3.1.3 Initialization**

None.

### **3.1.4 Higher-Layer Triggered Events**

None.

### **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 role in this protocol is to create the message types as discussed in Section 2, in addition to the core server behavior as discussed in [MS-OXCMSG].

## **4 Protocol Examples**

### **4.1 Playing a Voice Message**

Both the examples that follow assume a voice message has been stored by the server as described in section 2.

#### **4.1.1 Down-Level Experience**

A client consults the configuration information described in section 2.2 and sees that the outlookFlags setting indicates the client SHOULD provide a down-level experience for the voice message object it is about to display.

To provide the down-level experience, the client renders the voice message with all the functionality it would give to a typical Message object, as described in [MS-OXOMSG]. In particular, it allows the user to access the audio attachment included in the message using the standard mechanism provided by the client for accessing attachments.

Having accessed the content of the audio attachment file, the user uses an audio player application on his local computer that supports the attachment codec to play the audio content.

#### 4.1.2 Up-Level Experience

A client consults the configuration information described in section 2.2 and sees that the outlookFlags setting indicates the client SHOULD provide an up-level experience.

The up-level experience of the client includes the ability to click a single “Play” button and hear all audio attachments on the message played in the reverse order in which the attachments were added. The user presses this button and the client consults the attachment order information on the message (section 2.1.2.3) and sees the value is “vm2.wma;vm1.wma”. From this, it knows there MUST be two attachments on the voice message object with **PidTagAttachLongFileName** properties “vm2.wma” and “vm1.wma”, respectively.

The client downloads the attachment named “vm2.wma” and uses an audio player on the user’s local computer to play the WMA 9 Voice audio content; it knows the attachment is encoded with WMA 9 Voice because the **PidTagAttachMimeType** value of the attachment is “audio/wma”. After the audio finishes playing, the client downloads “vm1.wma” and plays it in a similar way.

The client up-level experience of the client application also includes the ability to read and edit audio notes directly on the voice message and the end user decides to use the feature. The client provides an editable area on the screen into which the user can type a set of notes. When the user is finished, the client persists the entered text in the **PidNameAudioNotes** property of the voice message object. The next time the user views this particular voice message object, he sees the notes he typed in because the client displays the content of the **PidNameAudioNotes** property of the voice message object.

## 5 Security

### 5.1 Security Considerations for Implementers

There are no special security considerations specific to the Voice Mail and Fax Objects protocol. General security considerations pertaining to the underlying transport apply (see [MS-OXCMSG]).

### 5.2 Index of Security Parameters

None.

## 6 Appendix A: Office/Exchange Behavior

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

- Office 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.

Preliminary

## 7 Index

Applicability statement, 6  
Client details, 10  
Examples, 11  
Fields, vendor-extensible, 6  
Glossary, 4  
Index of security parameters, 12  
Informative references, 6  
Messages, 7  
    Transport, 7  
    UI configuration, 9  
Normative references, 5  
Office/Exchange behavior, 12  
Overview, 6  
Preconditions, 6  
Prerequisites, 6  
Protocol details, 10  
    Client details, 10  
    Server details, 11  
Protocol examples, 11  
References, 5  
    Informative references, 6  
    Normative references, 5  
Relationship to other protocols, 6  
Security, 12  
    Considerations for implementers, 12  
    Index of security parameters, 12  
Security considerations for implementers, 12  
Server details, 11  
Standards assignments, 6  
Transport, 7  
UI configuration, 9  
Vendor-extensible fields, 6  
Versioning and capability negotiation, 6