Mail X

Mail eXtension: OCX-32 International Release v1.60e

What is new in International Release v1.60e?

Overview:

General Information
Installation
Introduction
Distribution
Technical Support
Binding
Right Button "Information"
License Information
Registering
Mail X Custom Address Support
FAQ
About Mail X
Mail Systems Supported
Mail X for Borland Delphi 1.0 & 2.0
Mail X for Visual Tools (MS Access 7.0, Visual FoxPro 3.0 & VC++ 4.0)

NEW!
Mail eXtension VIM GATEWAY Sessions

NEW!
Mail eXtension **Drivers & MS Office**

NEW!
Mail eXtension **Internet Driver RELEASE**

NEW!
Mail eXtension **International Version**

***IMPORTANT*: Mail eXtension OCX-32 & VIM 32 Libraries**

MailX Custom Controls:

Form Control
Session Control
Message Control
Recipient Control
File Attachment Control

MailX for Visual Basic:

Visual Basic Global Const

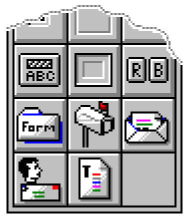
MailX .INI Settings:

Mail eXtension INI Setting for VIM & SMTP drivers

MailX for Borland Delphi:

Mail eXtension Component

General Information

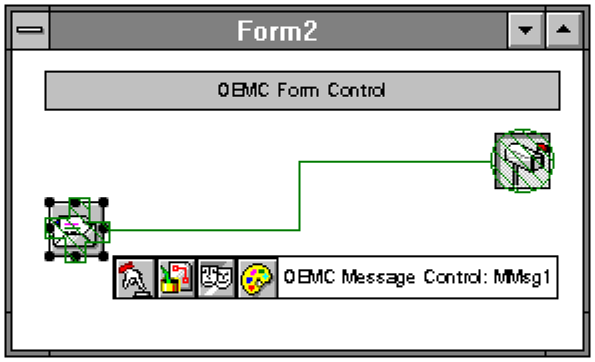


Mail eXtension is a SET of Visual Components that will ADD powerful e-mail capabilities to your Visual Basic, MS Access, Visual FoxPro, & Visual C++ Applications in minutes!!!. Mail X is a real Object Oriented Control based in the Microsoft Mail API (MAPI).

Mail X works with different *Mail Systems* without changing your code:

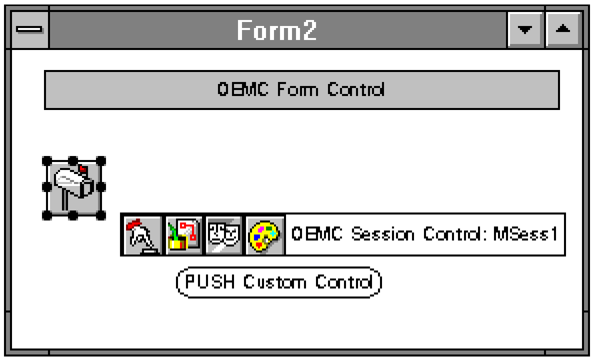
Your **Visual Applications** will use Microsoft Mail, Microsoft Exchange, cc:Mail, Lotus Notes & SMTP/POP3 protocols

Mail eXtension introduce a powerful Binding capabilities that will reduce the amount of code necessary with other Mail Controls.



Binding Mail X custom Control

INTERACTIVE MAILX:
Click the right button of your mouse and the Mail X Object Oriented Custom Control will Display all the information you need!!!.



Right Click Information

Mail eXtension can be used with:

- Visual Basic 4.0
- MS Access 7.0, Visual C++ 4.0 & Visual FoxPro (OCX-32 Version)

More Information.... Contact us at:

CompuServe:	CIS 73000,1661
Internet:	73000.1661@compuserve.com

Installation

Installing MAIL X is simple!!!,

Run the SETUP.EXE program available with the DEMO package. This Installation program will run only under Windows 95, Windows NT 3.51 or later

NOTE: *Be sure you have installed your Mail Client libraries (MAPI.DLL or VIM.DLL). You will need the **32 Bit version** of VIM libraries available at the COMPUSERVE/LOTUSM forum when using **MAILX32.OCX***

CANT LOAD MAILX CUSTOM CONTROL?
If you can't load MAILX in Visual Basic, make sure you have the MAILX32.LIC, MAILX32.OCX library in your PATH!!!. MAVIM32X.DLL library is required for VIM sessions.

CAN'T LOGON INTO YOUR MAIL SYSTEM?
If Mail X can't logon into your MAIL system, make sure you have the Mail Libraries (VIM32 or MAPI32) into your PATH directory.

Distribution

Mail X is royalty free after registering it. You can distribute it with any application that you develop with it.

One license is required for each developer using MAILX. Contact us for incredible quantity discounts!!!

REDISTRIBUTABLE COMPONENTS:

1- Mail eXtension OCX-32 version

- MAILX32.OCX	Mail eXtension OCX 32 version
- MAVIM32X.DLL	Mail eXtension 32 VIM router
- MAPSM32X.DLL	Mail eXtension 32 SMTP/POP3 driver

Registering Mail eXtension

How can I register Mail eXtension Components?

When you register MailX, you will receive the PASSWORD to decrypt the FULL version of MAILX.

-**CompuServe**: GO SWREG, ID: 5762 (U.S. $229) (**ONLY**)

Why you should register Mail eXtension today?

1.- FREE Upgrades. Mail X Registered users receive **FREE** upgrades. When registering Mail eXtension using On-Line registration, you will receive your personal passwords for future **FREE** upgrades. When new MAILX files are available (at Mail X Web Page) you will only need to download the new REGISTERED FILES and apply your PASSWORD to the new file.

2.- FREE Technical Support (via e-mail): Terckland Software offer **FREE** unlimited technical support via e-mail (only) to registered Users **ONLY**.

3.- Upgrade information: Notice of new versions, enhancements, and add-ins.

Technical Support

Mail eXtension support is provided by:

CompuServe:	73000,1661
Internet:	73000.1661@compuserve.com

Technical Support is **ONLY** available for Registered Users.

IMPORTANT: If you are an **UNREGISTERED USER,** We will **ONLY** reply messages for Installation problems.

When contacting us for Technical Support please use the application included with the package for this purpose OR send this information with your e-mail:

-Name:____________
-Serial Number: _________
-Company Name:___________
-Mail X Version Number:
-Mail X Version: OCX-16:__ OCX-32:__ VBX: __ VCL 16:__ VCL-32: __
-Mail System: cc:Mail: ___ Notes: ___ MS Mail___ Exchange____ Pop3/Smtp:___
-Operating System: WIN 3.1x: ___ Win 95: __ NT 3.51: ___ NT 4.0 ___

Binding MailX Custom Controls

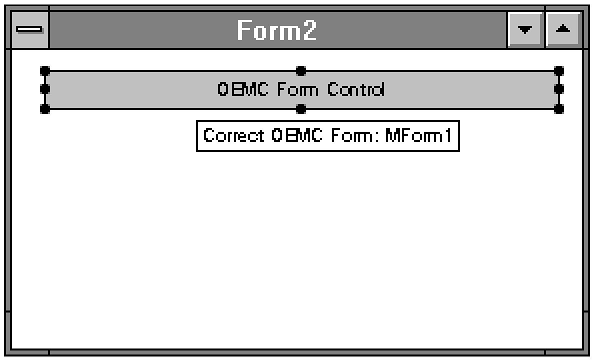
Mail X introduce a powerful binding support never seen in other Visual Basic Custom Controls.

BINDING is necessary to associate two or more Mail X Custom Controls!!!. (i.e.: if you want to read a Mail message in your inbox, you will need to associate or *bind* the MailX **Session Control** and the Mail X **Message Control**).

NOTE: To BIND Mail X Custom Control you need to identify your Visual Basic Form!!!. Add one (1) Mail X **Form Control** and your Application will be ready for the revolutionary Binding Support.

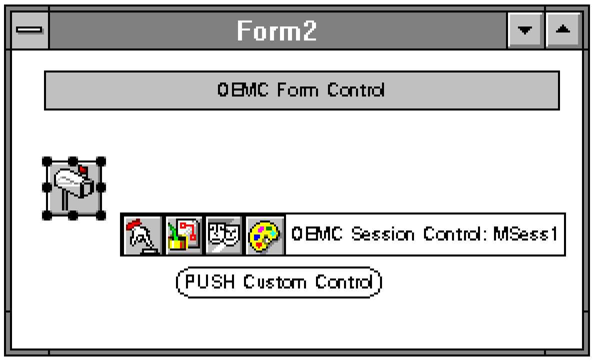
How to Bind Custom Control?

1. Be Sure to Include the Form Control in your Visual Basic Form that contains Mail X Custom Controls. (The Form Control is necessary to Add the Binding Support to Visual Basic)

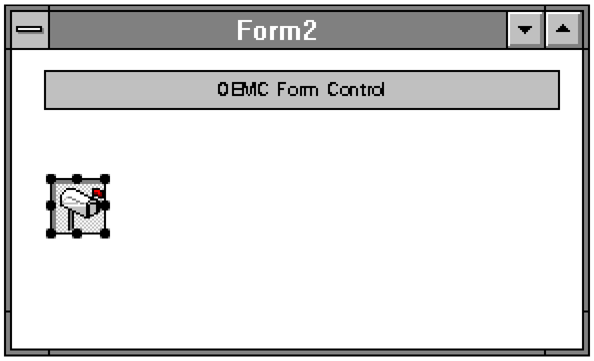


Form Control on your VB Form

2. Add and PUSH the custom control you want to bind with. To PUSH a Mail X Custom control you only need to select the Visual Object and Double click it with the right button of your mouse. You can also push a Custom control by Selecting the Push Button in the Help Label.



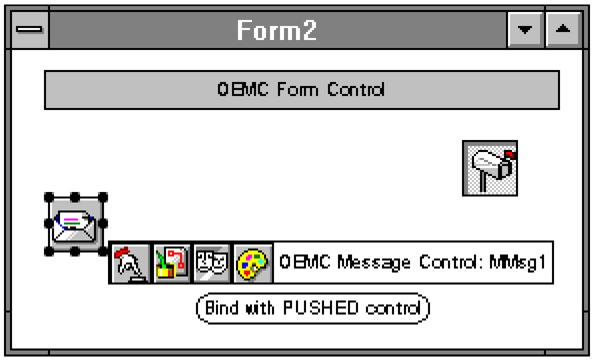
Add a Mail X Session Control



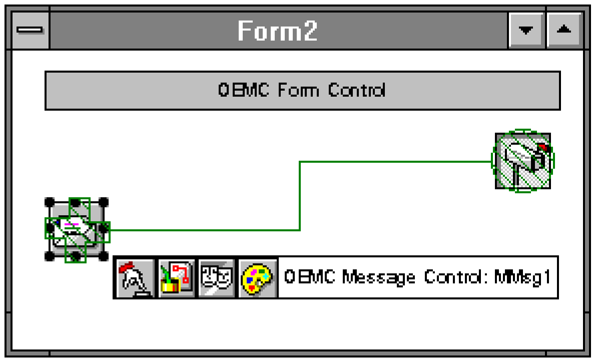
Pushed Session Control

NOTE: *When you Push a Custom Control it will look like a pressed button. The Help Label will Appear if you click the right button once over the custom control.*

3. Add the Second Mail X Custom Control and Bind it with the Pushed control. To Bind two custom control select the Second Mail X Custom Control and press <*SHIFT* + *right button of your mouse*>. You can also select the Bind Button available in the Help Label.



Binding Custom Controls



Connection drawing over your VB Form

When You Bind Mail X Custom Controls, a Connection is drawn over your Visual Basic Form!!!. When binding two Custom Control the Target BindString property is modified indicating*:*

BINDSTRING: **Formxxxx.*MailX_CustomControl***

The Formxxxx is the MXFormName property of the Mail X Form Custom Control, and the MAILX_CustomControl is the NAME of the associated control.

Note: *The* ***Formxxx*** *will only be added to your BindString Control property if the Binding Custom Controls resides on different Forms*.

To Connect Custom Control on different forms, be sure to include one Form Control over each Visual Basic Form.

OTHER BINDINGS:

In the same way you BIND a Message control with the associated SESSION control you can BIND:

- Recipient Controls with Message Controls
- File Attachment control with Message Controls

NOTE: Graphic Binding Support is only available for Visual Basic Applications. When using MS Access, Visual FoxPro 3.0 or Visual C++ 4.0 you will need to use code-binding support

Mail Systems supported

Mail X supports:

- Microsoft Mail or Exchange,

- cc:Mail or Lotus Notes (VIM Libraries are required)

- SMTP/POP3 Protocols (WINSOCK Libraries are required)

Mail X is able to open two or more sessions (*depending of your Mail System*) from different mail systems at the same time.

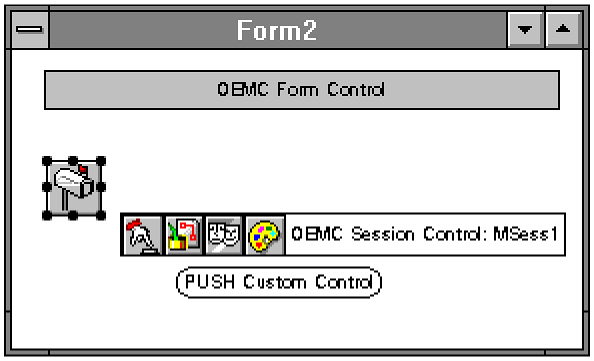
Mail X allows you to select the Mail System you want to work with at **Runtime** without changing your code!!!

Right Button Information

The RIGHT button of your mouse will help you to:

- Push MAILX Custom Controls
- Bind Custom Controls
- Change Control Properties
- Display Information About the selected custom control

When you select a MAILX Custom Control you will only need to "**CLICK**" the right button of your mouse over the control and the "*HELP LABEL*" will be displayed. The "help label" displays the name of the Custom Control and several button to: PUSH, BIND and Change properties of the selected control.



Help Label

Help Label

The Help Label is always available when you click the RIGHT BUTTON of your mouse!!!. With the Help Label, you can:

- Push,
- Bind, and
- Change the Mail X Control properties!!!

Push Control

Select The Mail X Custom Control you want to bind with.
The Custom Control currently selected looks like a pressed button.

Bind Control

Associate two or more Mail X Custom Control.
Bind connection is drawn over your VB Form.

Change Mail X Custom Properties

Change the control properties with a Custom Dialog Box.

About Mail X

Mail X has been developed by:

Terckland Software
August 1996

Custom Control Developers for Visual Basic and Delphi.

CompuServe:	73000,1661
Internet:	73000.1661@compuserve.com
	http://ourworld.compuserve.com/homepages/mailx

MAPI.DLL is distributed by Microsoft Corporation. MAPI.DLL & MAPI32.DLL are required to access the Microsoft Mail or Microsoft Exchange Server.

VIM.DLL & VIM32.DLL are distributed by Lotus Corporation. VIM.DLL & VIM32.DLL are required for MAILX to access the Mail Data information in your network!!!

NEW!

What is New in v.1.60?

Mail eXtension INTERNATIONAL RELEASE v1.60e is the second mayor update of our Mail Custom Controls for Visual Basic. This new version has been completely redesigned to enhance the VIM performance.

1. Full support for Visual Basic 4.0

Mail X v1.60 has been fully tested with the Enterprise Version of Visual Basic 4.0 for Windows. New 32 Bits OCX custom controls for VB4. Mail eXtension now supports **Microsoft Exchange Server** sessions.

*2. *New Mail eXtension International Release*

Mail X 32-Bits drivers are now available for English, German, Spanish, Italian, French, Norwegian & Icelandic. (VIM & SMTP/POP3 Drivers)

*3. *New Mail eXtension SMTP/POP3 Driver*

Mail X International Release 1.60e now includes the SMTP/POP3 BETA FINAL. The New Internet Driver now support BinHex, Base 64 & UUDecoder, Quoted-Printable *(Encoder/Decoder),* MIME encoder*,* Address Book Functions*,* Async 16 & 32 Bits Drivers, Internet Message Store, *etc*

*4. *New MS Access 7.0, Visual FoxPro & Visual C++4.0 support*

Mail X is now compatible with your favorite development tool!. Mail-enable all your Windows 95 Applications using Mail eXtension OCX-32 version 1.60

*5. *New Microsoft Exchange Extended Mapi Interface implementation (OCX-32 only)*

Mail X version 1.60 now implements an Extended Mapi Interface when using MS Exchange sessions. New spooler actions for sending & receiving messages, retrieve current user name, change passwords, invoke config dialog, profile wizards an so on!. (OCX-32 Version Only!)

*6. *New Enhanced VIM Support*

Mail X RELEASE v1.60 distributes the last version of MAVIM32X v.1.60. The MAPI interface has been extended to enhance the VIM performance over 400%, VIM category support. **Lotus Notes** sessions are now supported. **VIM GATEWAY sessions** are Now supported

*7. *New Mail eXtension VIM / SMTP Enhanced Address Book support*

Mail eXtension v1.60 now enhance VIM / SMTP Address Book User interface. New Find tool is available when using Address Book User Interface.

8. Mail eXtension Now supports

Mail eXtension v1.60 supports:
-Lotus Notes
-Lotus cc:Mail
-Microsoft Exchange Server
-Microsoft Mail
-Internet Sessions (POP3 / SMTP3 protocols)
-Custom Simple Mapi (SMAPI) Libraries.

Visual Basic Global Const (OCXMAILX.BAS)

```
' Mail X Session Constants
Global Const ERROR_LOADING_LIBRARY = 1
Global Const ERROR_NOT_CONNECTED = 2
Global Const ERROR_EX_FUNCTION = 3
Global Const SESSION_NONE = 0
Global Const SESSION_MSMAIL = 1
Global Const SESSION_VIM = 2
Global Const SESSION_POP3SMTP = 3
Global Const SESSION_CUSTOMLIB = 4

Global Const ACTION_INVOKE_CONFIG = 1
Global Const ACTION_FORCE_DOWNLOAD = 2
Global Const ACTION_FORCE_UPLOAD = 3
Global Const ACTION_INVOKE_WIZARD = 4

' Mail X Message Constants
Global Const ACTION_DELETEMAIL = 1
Global Const ACTION_FINDNEXT = 2
Global Const ACTION_FINDFIRST = 3
Global Const ACTION_CLEARMSG = 4
Global Const ACTION_CLEARRECIP = 5
Global Const ACTION_CLEARFILE = 6
Global Const ACTION_NEW = 7
Global Const ACTION_REPLY = 8
Global Const ACTION_FORWARD = 9
Global Const ACTION_REPLYALL = 10
Global Const ACTION_COPYMSG = 11
Global Const ACTION_SAVEMSG = 12
Global Const ACTION_SENDMSG = 13

Global Const ERROR_MSG_NOTCONNECTED = 1
Global Const ERROR_MSG_NOTFETCH = 2
Global Const ERROR_MSG_NOT_BOUND = 3
Global Const ERROR_MSG_SENDMAIL = 4
Global Const ERROR_MSG_SAVEMSG = 5
Global Const ERROR_MSG_EMPTYMSG = 6
Global Const ERROR_MSG_DELETE = 7
Global Const ERROR_MSG_SETMSGID = 8
Global Const ERROR_MSG_FINDNEXT = 9
Global Const ERROR_MSG_FINDFIRST = 10
Global Const ERROR_MSG_FETCHMSG = 11
Global Const ERROR_MSG_EX_FUNCTION = 13

Global Const INBOX_MSG = 0
Global Const COMPOSE_MSG = 1

Global Const NEW_MSG = 0
Global Const REPLY_MSG = 1
Global Const FORWARD_MSG = 2
Global Const REPLYALL_MSG = 3

Global Const CLEAR_MSG = 0
Global Const CLEAR_RECIP = 1
Global Const CLEAR_FILE = 2

' Mail X Recipient Constants
Global Const ACTION_DETAILS = 1
Global Const ACTION_ADDRESS = 2
Global Const ACTION_ADDRECIPIENT = 3
Global Const ACTION_DEL_RECIPIENT = 4
Global Const ACTION_RECIP_SET = 5
Global Const ACTION_INSERTCUSTOM = 6
Global Const ACTION_ORIG_SET = 7
Global Const CLASS_ORIG = 0
Global Const CLASS_TO = 1
Global Const CLASS_CC = 2
Global Const CLASS_BCC = 3

Global Const MAPI_ORIG = 0
Global Const MAPI_TO = 1
Global Const MAPI_CC = 2
Global Const MAPI_BCC = 3

Global Const LIST_ONLY = 0
Global Const EDIT_TO = 1
Global Const EDIT_TO_CC = 2
Global Const EDIT_TO_CC_BCC = 3

Global Const ERROR_REC_NOTBOUND = 1
Global Const ERROR_REC_MSGNOT_BOUND = 2
Global Const ERROR_REC_EMPTY = 3
Global Const ERROR_REC_SESS_NOTCONNECT = 4
Global Const ERROR_REC_BADCONTROL = 5
Global Const ERROR_REC_AMBIGUOUS = 7

Global Const ORIGINATOR = 0
Global Const MSG_RECIPIENTS = 1
```

' Mail X FILE Constants

```
Global Const ACTION_DELETEFILE = 1
Global Const ACTION_REMOVEFILE = 2
Global Const ACTION_ADDFILE = 3
Global Const ACTION_FILESET = 4
Global Const ACTION_DELETE_TEMPFILE = 5

Global Const ERROR_DELETE_TEMP = 1
Global Const ERROR_FILE_EMPTY = 2
Global Const ERROR_FILE_NOT_BOUND = 3
Global Const ERROR_MSGNOT_BOUND = 4
Global Const ERROR_FILE_BADCONTROL = 5
Global Const ERROR_FILE_UNABLEADD = 7

Global Const NO_FLAGS = 0
Global Const MAPI_OLE = 1
Global Const MAPI_OLE_STATIC = 2
```

VIM 32 Libraries & Mail X OCX 32

Mail X OCX 32 requires the VIM 32 libraries distributed by LOTUS.

Updated Lotus cc:Mail VIM 32 libraries can be found in:

CompuServe:	GO LOTUSM	VIM Library
INTERNET:	FTP:	ftp.support.lotus.com

Copy the Files: VIM32.DLL, MEDB632.DLL & CHRSET32.DLL in your ..\SYSTEM directory.

NOTE: Mail X OCX 32 doesnt work with the 16 bit VIM libraries. Youll need to update your VIM libraries with the VIM 32 files.

Mail eXtension ObjRef Property

OCX-32 Users:

Description:

Use this Property to retrieve the Object reference when using code-binding

Usage:

[form].MXFile1. **BindWith (***[form].MXMessage1.* **ObjRef)**
[form].MXRecipient1. **BindWith (***[form].MXMessage1.* **ObjRef)**
[form].MXMessage1. **BindWith (***[form].MXSession1.* **ObjRef)**

Remarks:

Runtime only.
This Property is and is ONLY available for OCX-32 users

Data Type:

Long

Borland Delphi Users:

Description:

This property Copy or Fetch Mail eXtension Message, Recipient or File components

Usage:

[form].MXFile1. **ObjRef** =*[form].MXMessage1.* **ObjRef**	**{Fetch action}**
[form].MXFile1. **ObjRef** =*[form].MXFile2.* **ObjRef**	**{Copy action}**
[form].MXRecipient1. **ObjRef** =*[form].MXMessage1.* **ObjRef**	**{Fetch action}**
[form].MXMessage1. **ObjRef** =*[form].MXMessage2.* **ObjRef**	**{Copy action}**

Remarks:

Runtime only.
This Property is and is ONLY available for Borland DELPHI.
Visual Basic Developer can use the = operator:

[form].MFile1 =[form].MMess1	{Fetch action}
[form].MFile1=[form].MFile2	{Copy action}
[form].MReci1 =[form].MMess1	{Fetch action}
[form].MMess1 =[form].MMess2	{Copy action}

Data Type:

Pointer

BindWith Method

Description:
This Method binds a Mail eXtension Component at Runtime. Use this method when using MS Visual FoxPro, MS Access or Visual C++ 4.0

Usage:
[form].MXFile1. **BindWith (***[form].MXMessage1.* **ObjRef)**
[form].MXRecipient1. **BindWith (***[form].MXMessage1.* **ObjRef)**
[form].MXMessage1. **BindWith (***[form].MXSession1.* **ObjRef)**

Remarks:
Only available when using Mail eXtension OCX -32 version.

Mail eXtension OCX-32 Version 1.60

Terckland Programming
Windows solutions

Mail eXtension for Microsoft Access 7.0, Visual FoxPro & Visual C++ 4.0.

Mail eXtension OCX-32 Version 1.60 NOW supports the most popular Visual Development Tools!. Mail-enable ALL your Windows 95 applications using Mail extension OCX-32

What is Code-Binding?

Code-binding associate Mail eXtension components at Runtime.
Visual binding support is only available for Visual Basic & Delphi programmers, you will need to use code-binding when using another Visual development tool.

No Mail eXtension Form Control is required when using Code-Binding.

How to implement Code-Binding?

Because MailX Form Control is not required when using code-binding, you will **only** need to add MailX Session, Message, Recipient & File Controls at design time. When loading your Visual Form, you will need to use the ObjRef Property and BindWith Method in order to associate your Mail eXtension Controls at Runtime.

Microsoft Access 7.0 Example:

```
Private Sub Form_Load()
        MMsg1.BindWith (MSession1.ObjRef)
        MReci1.BindWith (MMsg1.ObjRef)
End Sub
```

Microsoft Visual C++ 4.0 Example:

```
BOOL CFormDlg::OnInitDialog()
{
        CDialog::OnInitDialog();
        .
        .
        .
        // TODO: Add your Mail eXtension Code-Binding support here
        m_MMessage.BindWith(m_MSession.GetObjRef());

        return TRUE;   // return TRUE   unless you set the focus to a control
}
```

Microsoft Visual FoxPro Example:

```
CLEAR
frmMyForm = CREATEOBJECT("FormMail")
frmMyForm.Closable = .F.   && Disable the Control menu box

*-- Bind the controls
frmMyForm.MailMess.BindWith(frmMyForm.MailSess.ObjRef)
frmMyForm.MailSess.Mail_Type=2
frmMyForm.SHOW   && Display the form
READ EVENTS   && Start event processing


DEFINE CLASS FormMail AS FORM
```

```
        ADD OBJECT MailMess AS OleControl WITH ;
                OleClass = 'MailX.MMsgCtrl'

        ADD OBJECT MailSess AS OleControl WITH ;
                OleClass = 'MailX.MSessCtrl'

        ADD OBJECT QuitButton AS QuitBtn WITH ;
                Name = "QuitButton"

        ADD OBJECT SignOn AS SignOnBtn WITH ;
                Name = "SignOn"

        ADD OBJECT MsgCount AS MsgCountBtn WITH ;
                Name = "MsgCount"

ENDDEFINE

DEFINE CLASS QuitBtn AS CommandButton   && Create Command button
        Caption = '\<Quit'   && Caption on the Command button
        Cancel = .T.   && Default Cancel Command button (Esc)
        Left = 17   && Command button column
        Top = 0   && Command button row
        Width = 120
        Height = 44   && Command button height
        PROCEDURE Click
                CLEAR EVENTS   && Stop event processing, close Form
ENDDEFINE

DEFINE CLASS SignOnBtn AS CommandButton   && Create Command button
        Caption = "Sign On"   && Caption on the Command button
        Cancel = .T.   && Default Cancel Command button (Esc)
        Left = 17   && Command button column
        Width = 120
        Top = 50   && Command button row
        Height = 44   && Command button height
        PROCEDURE Click
        ThisForm.MailSess.Logon=.T.
ENDDEFINE

DEFINE CLASS MsgCountBtn AS CommandButton   && Create Command button
        Caption = "MsgCount"   && Caption on the Command button
        Cancel = .T.   && Default Cancel Command button (Esc)
        Left = 17   && Command button column
        Width = 120
        Top = 100   && Command button row
        Height = 44   && Command button height
        PROCEDURE Click
        nCount=ThisForm.MailMess.MsgCount
        if ThisForm.MailMess.ErrorNum=0 then
                nRes = MESSAGEBOX('MessageCount='+STR(nCount),0,'Message Count')
        endif
ENDDEFINE
```

Mail X Drivers & MS Office

Terckland Programming
Windows solutions

Mail eXtension Drivers Version 1.61

The most powerful Mail Drivers for Visual Tools are now 100% compatibles with your MS Office applications.

Mail eXtension Drivers 100% SMAPI Compatibles. Version 1.61

New Mail X Drivers:
VIM Driver Version 1.61.00 (16 & 32Bits)
SMTP/POP3 Driver PRE-RELEASE # 4 (16 & 32Bits)

now allow you to send messages from any MS Office or even any SMAPI compatible application to the Internet/VIM using the New Mail X Drivers!

You will only need to apply these changes to your Windows workstation:

- 16 Bits System

Include the following keyword in your WIN.INI:
[Mail]
MAPIDLL=C:\WINDOWS\SYSTEM\MAPSM16X.DLL (if you are using Internet)
or
MAPIDLL=C:\WINDOWS\SYSTEM\MAVIM16X.DLL (if you are using VIM)

- 32 Bits Systems

Include this value into your Windows 95/NT Registry Database:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft Office\95\Mapidll\DLL32 = MAPSM32X.DLL
or
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft Office\95\Mapidll\DLL32 = MAVIM32X.DLL

(Use the Registry Editor)

Terckland Software
August 1996

Mail eXtension for Visual Basic & Delphi

Terckland Programming
Windows solutions

Mail eXtension for Visual Basic 4.0.

Different messaging systems are offered by several vendors, and a wide range of applications have been built to use them. But each of these messaging systems have different programming interfaces, making it difficult for one application to interact with several of them at once. Until now, visual developers looking for a messaging custom controls had few choices. VBX-based controls provide basic e-mail capabilities, but are limited to a specific LAN-based system and dont support the new Visual programming tools standards.

Mail eXtension is e-mail custom controls finally done right. It provide the essential e-mail capabilities to mission-critical applications and is compatible with several Mail Systems. Mail eXtension introduce a solid messaging features that enable your applications to receive and send e-mails in minutes.

Mail eXtension architecture is a dramatic boost over current Visual Basic Custom Controls. Introducing its new binding capabilities, Visual Basic & Delphi developers will associate mail objects in seconds. In addition, Mail eXtension supports the most popular visual programming tools:

1- OCX-32 for Microsoft Visual Basic 4.0, MS Access 7.0, Visual FoxPro & Visual C++ 4.0

Mail Systems

Mail eXtension applications can be run on 32-bit platforms available on Windows 95 and Windows NT. Mail eXtension-compliant application communicate with any messaging system as long as the appropriate drivers are installed.
Our Mail API eXtension also boots the performance of different LAN-based systems.
Mail eXtension is compatible with most important Messaging Systems. Innovative Mail applications can be created combining Visual Programming   with Lotus cc:Mail, Lotus Notes, Microsoft Exchange, Microsoft Mail, SMTP/POP3 protocols.

How Mail eXtension works

Mail eXtension use a modular architecture. At one end is the Client application using the Mail eXtension components. Extension drivers sit between the unique messaging systems and the SMAPI standard. At the Back-end is the specific programming interface for different messaging systems.

Easy and Powerful binding

To reduce the time needed to associate Visual Component, Mail eXtension introduce a powerful graphic binding support. Developers will only need to push & click custom controls when designing e-mail enabled Applications.

Distribution

Mail eXtension is Royalty Free distribution after registering it.

Mail eXtension is the perfect component for your Visual Basic applications.

For More information

Download your Mail eXtension DEMO:

-via CompuServe:
GO LOTUSM
VIM Library / MAILX61.EXE

-via FTP:
FTP FTP.WINSITE.COM
pub/pc/win95/programr/vbasic
MAILX6.ZIP

CALL: +58 (2) 976.7070

Mail Extension

Version Release 1.60e (International Version)
Terckland Programming,
E-Mail: 73000.1661@compuserve.com
Price: U.S. $ 229

Versions: OCX-32.

Platform: Windows 95 & NT.
Messaging System: Microsoft Mail, Microsoft Exchange, Lotus cc:Mail, Lotus Notes & SMTP / POP3 protocols

Terckland Programming
E-Mail: 73000.1661@compuserve.com

Mail eXtension Questions & Answers
Terckland
July 19, 1996

Q. Is there a VBX-version of Mail eXtension?
A. Yes, Terckland has included the Mail eXtension VBX Version in the MAIL X FULL DEMO package. Visual Basic 3.0 & Delphi 1.0 examples are also included

Q. What files do I need for Mail eXtension?
A. Mail eXtension requires the MAPI32.DLL & VIM32.DLL in order to establish a mail session with your Mail Server or Post Office

Q. The Mail eXtension Session Control returns Error Loading Mail Library?
A. The MAIL X Session control will return this error if your MAPI32.DLL or VIM32.DLL can not be found in your workstation PATH. Please include the Mail library path in your AUTOEXEC.BAT

Q. Is the Mail eXtension source code available for registered users?
A. Mail eXtension Source Code is not currently available

Q. How can I send messages to an external recipient address?
A. Mail eXtension v1.60e now supports Custom Address. Mail eXtension Custom Address are useful when sending messages to any gateway (*or remote PostOffice*) defined in your Mail Server. You only have to specify your Recipient Address and Mail eXtension will try to send the e-mail through your PostOffice.

Q. What Mail eXtension files should I distribute with my application?
A. You should only distribute your MAILX32.OCX version and the MAVIM32X.DLL library

Q. Why I cant run multiple instance of my application when using Lotus Notes R3?
A. The Lotus Notes VIM libraries only shared access to sessions across multiple threads, but not across multiple processes.

Q. I have problems when deleting messages using Lotus Notes
A. When Deleting a Mail Message, You will have to re-scan" your active messages because Lotus Notes automatically re-indexes message references.

Q. I have Lotus Notes & Lotus cc:Mail in my workstation. How can I choose between them?
A. The Mail eXtension VIM driver will Load the first VIM32.DLL available in your PATH. Nevertheless you can try to load any specific version of VIM32.DLL using your WIN.INI file:
- INCLUDE in your WIN.INI File the following section

```
[MAILX]
VIM32=M:\CCDATA       (Set your VIM32.DLL path)
```

Q. I have MS Mail & MS Exchange in my workstation. How can I choose between them?
A. The Mail eXtension Component will Load the first MAPI32.DLL available in your PATH. Nevertheless you can try to load any specific version of MAPI32.DLL using the Custom Library Type

Q. Can I use Mail eXtension with MS Access, Visual C++ or Visual FoxPro?
A. Yes, Mail eXtension OCX-32 Version 1.60e NOW Support the most popular Visual Development Tools. Because visual binding is only available for Visual Basic & Delphi programmers, you will need to

use code-binding with your applications.

Q. How can I set the Default Profile Name when using MS Exchange?
A. Use the MAILX Session User Property to set the Default Profile Name.

Q. How can I set the Internet Address Book File?
A. Use the MAILX section in your WIN.INI file:

```
[MAILX]
Internet AB Book=C:\MAILXDLL\Book.abx
```

Q. How can I set different TEMP directory when using the VIM/SMTP Driver?
A. Use the MAILX section in your WIN.INI file:

```
[MAILX]
TEMPDIR=C:\TEMP\
```

Q. How can I activate VIM Gateway Sessions?
A. Use the MAILX section in your WIN.INI file:

```
 [MAILX]
VIM_GATEWAY=1
```

Q. Can I remove the Mail X Internet message Store?
A. Yes, you can delete the ..\TEMP\INBOX.FLD directory and Mail X SMTP/POP3 driver will prompt you to create a new message store when creating a new Internet Session

Q. How can I reset the Internet message Store?
A. Delete the ..\TEMP\INBOX.FLD directory and Mail X SMTP/POP3 driver will prompt you to create a new message store when creating a new Internet Session

Mail eXtension & Internet Support

Terckland Programming
Windows solutions

Mail eXtension Driver for SMTP/POP3 protocols.

Mail eXtension Version R1.60e now introduce the **Internet Driver RELEASE Version**
Internet Driver RELEASE sends & receive messages using SMTP/POP3 protocols.

Now features:

-UUDecoder
-Enhanced BinHex decoder
-Quoted-Printable (Encoder/Decoder)
-MIME encoder (base 64),
-Address Book functions,
-Internet Message Store,
-POP3 protocol,
-MIME decoder (base64),
-File attachment & Recipient handling
-New Messages arrived method

How to use the Mail eXtension Internet Driver?

When you Sign On the Internet Driver, set the User Name to your POP3 Account **(NOTE: do not include your domain name. I.e.: If your POP account is mailx@terckland.com,' your correct user name is mailx).**

How to configure your Internet Servers?

When you Sign On the Internet Driver, Click over the Host configuration image in order to set the correct IP address for your SMTP & POP3 Servers.

Is there any special property for the Internet Driver?

Yes, When using the Internet Driver, you can use the Session Properties: **ReturnAddress, SMTPServer, POP3Server, RealName & LeaveMailOnServer**

Mail eXtension Web Pages:!

The Latest SMTP/POP3 Driver can be downloaded from:

http://ourworld.compuserve.com/homepages/mailx/betatest.htm

Terckland Software
August 1996

Mail eXtension License Information

Terckland Programming
Windows solutions

END-USER LICENSE AGREEMENT FOR MAIL EXTENSION COMPONENTS

By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this Agreement, you are not authorized to use the SOFTWARE PRODUCT.

1. **GRANT OF LICENSE**. This EULA grants you the following rights:
   Use MAIL EXTENSION in your development environment. Freely distribute it as a RUNTIME component of your applications. The author asks for NO royalties or run-time fees. Also, this help file and accompanying demos do not have to be included in your distribution package. You may also make copies of the SOFTWARE PRODUCT for backup and archival purposes.

2. **RESTRICTIONS**.
   --You may not reverse engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.
   --You may not rent or lease the SOFTWARE PRODUCT.
   --Modify MAIL EXTENSION, and this help file in any way.
   --Remove any proprietary notices, labels, or marks on the program and accompanying documentation (help file).
   --Sell or resell MAIL EXTENSION and/or any of its accompanying products and documentation, by itself or as part of another package, except as a RUNTIME component of another application whose primary purpose is NOT illustration of the functionality of MAIL EXTENSION.

3. **TERMINATION**. Your rights under this EULA terminate upon the termination of your Mail eXtension EULA, or without prejudice to any other rights, TERCKLAND may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE PRODUCT.

4. **LIMITATION OF LIABILITY**. The author of MAIL EXTENSION makes no warranties, expressed or implied, in relation to this product. In no event shall the author be liable to you for any damages, including any loss of profits, loss of data, including but not limited to special, incidental, consequential, or indirect damages arising from the use of this software.

Mail eXtension INI settings

Mail eXtension VIM & SMTP drivers store configuration information in your WIN.INI file under the MAILX section:

Internet AB Book	Use this Keyword to set the Path and File name of your Internet Address Book	SMTP/POP3
POPSERVER	Set your POP3 server name	SMTP/POP3
SMTPSERVER	Set your SMTP server name	SMTP/POP3
TEMPDIR	Set the TEMP directory for your File attachment storage	SMTP/POP3 & VIM
Internet MsgStore	Set the Internet Message Store PATH	SMTP/POP3
SMTP WordWrap	Set the Default WORD-WRAP status	SMTP/POP3
SMTP Quoted-Printable	Set Default QUOTED-PRINTABLE property	SMTP/POP3
LEAVEMAILONSERVER	Set the LeaveMailOnServer default status	SMTP/POP3
VIM32	Set the Default VIM32.DLL library Path	VIM
VIM16	Set the Default VIM.DLL library Path (16 Bits version)	VIM
NOTES_ALL_BOOKS	Set This Property to one (1) to indicate Lotus Notes session to search All Address Books available when Resolving recipients	VIM
VIMSEL_ENCRYPT	Set to one (1) to indicate Lotus Notes Session to Encrypt messages	VIM
VIMSEL_SIGN	Set to one (1) to indicate Lotus Notes Session to Sign messages	VIM
VIMSEL_DELIVERY_REPORT	Set to one (1) to indicate Lotus Notes Session to receive Delivery reports	VIM
VIMSEL_NONDELIVERY_REPORT	Set to one (1) to indicate Lotus Notes Session to receive non delivery reports	VIM
VIM_GATEWAY	Set to one (1) to enable Lotus cc:Mail/Notes Gateway Sessions	VIM

Mail eXtension VIM Gateway Session

Mail eXtension International **Release 1.60e** NOW support Lotus cc:Mail GATEWAY Session!.

When using Gateway Session you can Sign On your Post Office as a PO User and Set the FROM or Originator Header when sending messages.

To enable Gateway Session SET the clause VIM_GATEWAY in your WIN.INI file:

```
[MAILX]
VIM_GATEWAY=1
```

When you Sign On set the **USER Property** as the PostOffice you are logging in as (not the PostOffice you are logging into)
Before Sending the Message SET the FROM clause using the NEW Recipient Action: **ACTION_ORIG_SET**

NOTE: DO NOT set this option if you are not sure about gateway Sessions. If you TURN ON this option, you will NOT be able to Sign On as a User.

Terckland Software
August 1996

Mail eXtension International Release

Terckland Programming
Windows solutions

Mail eXtension International Release

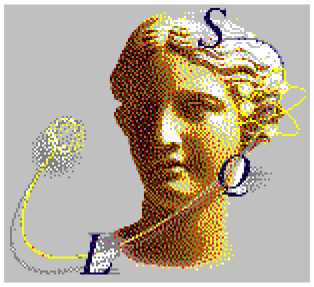
The Most Powerful Mail Control for Visual Tools is now available for your native language. The New Mail eXtension 32 Bits Drivers (**International Release 1.60e)** NOW include MULTI-LANGUAGE support for Windows 95 & Windows NT 3.51 or later…

Mail X 32-Bits Drivers (**VIM & SMTP/POP3 protocols**) are now available:

- **-English,**
- **-Italian,**
- **-German,**
- **-Spanish,**
- **-French,**
- **-Norwegian &**
- **-Icelandic**

Set your native Language using your Windows 95 / Windows NT controls Panel and Mail X Drivers will display the appropriate resource for your language!.

Terckland Software
August 1996



Mail eXtension Component for Borland Delphi 1.0 & 2.0

Mail eXtension RELEASE v1.60 introduce the MailX DCU for Borland Delphi 1.0 & 2.0

Files: ..\DELPHI1\MXMAILX.DCU (Mail eXtension for Delphi 1.0)
..\DELPHI20\MXMAILX.DCU (Mail eXtension for Delphi 2.0)
..\HELP\MAILX.HLP (Mail eXtension Help File)
..\HELP\MAILX.KWF (Mail eXtension Keyword File)

Installation:

1. Create a directory for the Mail eXtension Component.
2. Start up Borland Delphi
3. Select Options | Install Components
4. Install the Component using the MXMAILX.DCU file
5. Exit Delphi. (Do not save any changes to project, if asked).
6. Move the .hlp and .kwf files to the \delphi\bin directory.
7. Run the 'helpinst' program found in '.....\delphi\help'
8. Open File '......\delphi\bin\delphi.hdx'
9. Add the new keyword file (.kwf extension) found in the \delphi\bin directory
10. Select File|Save, then Exit
11. The component will be installed in the Mail eXtension tab and the help file will be active

Form Control

Properties

Overview:

The Mail X Form Control is required to enable the Visual binding capabilities of Mail X Custom Controls.

You should ADD only one (1) Form Control to each Visual Basic or Delphi FORM containing other Mail X Custom Controls.

Note: All Mail X Custom Controls will be referenced by the NAME associated to the Form Control. The Name of the Form Control is stored in the MXFormName property.

MXFormName Property

Description:
The MXFormName Property contains the Name associated with your Control. By Default, MAILX will name your Form Control: **MXFormName**: FormTag*x*.

You can change your MXFormName Property at Design Time (*It should never be done at runtime because associated MailX Custom Control will not be bound!)!*.When Changing the MXFormName property, be sure to Re-Bind your Mail X Custom Controls associated with your Form.

Usage:
FormTagName$=[*form*].*MForm*.**MXFormName**

Data Type:
String

Properties:

- About
- BackColor
- Height
- Index
- Left
- MXFormName
- Name
- Tag
- Top
- Width

Session Control

Properties Event Methods

Overview:

The Session control is the principal object to access your Mail System. Mail X allows to establish different sessions simultaneously *(depending of your mail system)* without changing your code. You just need to select your Mail System when designing your application. Mail X is also able to select the Mail System at runtime.

The Session Control Object ONLY handle your connection to your Mail Server, you will need to bind a Message Control to handle your mail messages available in your inbox.

When using MS Exchange, use the User property to set the default profile name.

Click your Right button and change the Session Control properties with a Mail X Session Control DialogBox

Note: You must have a Mail X Form Control before adding Mail X Session Controls on your Visual Basic Form

Action

Description:
This property set a new Action in your Mail X Session Custom Control.

Usage:
[form].MSess. **Action**=i_NewAction

Values:

Action ID	Number	Description
ACTION_INVOKE_CONFIG	1	Invokes the Session Configuration Dialog.Not all Mail eXtension Drivers supports this action
ACTION_FORCE_DOWNLOAD	2	Force spooler to retrieve new messages from Transport service providers (ONLY available for Extended Mapi drivers)
ACTION_FORCE_UPLOAD	3	Force spooler to send outgoing messages to all Transport service providers (ONLY available for Extended Mapi drivers)
ACTION_INVOKE_WIZARD	4	Show the Profile Wizard DialogBox (ONLY available for Extended Mapi drivers)

Remarks:
This Property is runtime write only. An Error event will be generated if the action fails.

Data Type:
Integer

ChangePassword

Example

Description:

This property sets a new password that will be used when you logon your Mail system.

Usage:

[*form*].*MSess.***ChangePassword**= *sz_NewUserPassword$*

Remarks:

You must set a valid string to change your session password, otherwise, an error will be generated. Note: Not all Mail eXtension Drivers supports this Property

Data Type:

String

CurrentFolder

Example

Description:

This property sets and returns the current folder or category that will be used when you scan your messages.

Usage:

sz_CurFolderName$=[*form*].*MSess.***CurrentFolder**
[*form*].*MSess.***CurrentFolder** = *sz_NewFolderName$*

Remarks:

You must set a valid FolderName; otherwise you will not be able to read to your messages. This New Property is only available with the VIM Driver (cc:Mail). Set CurrentFolder= (empty string), when using the Inbox

Data Type:

String

CurrentUser

Description:
This property returns the Current User Name

Usage:
sz_CurUserName$=[*form*].*MSess.***CurrentUser**

Remarks:
This New Property is only available with the VIM Driver (cc:Mail).

Data Type:
String

DefaultPath

Example

Description:
This property sets and returns the DefaultPath used when you access your Mail Server.

Usage:
sz_DefaultPath=[*form*].*MSess.***DefaultPath**
[*form*].*MSess.***DefaultPath**= *sz_DefaultPath*

Remarks:
This New Property is only available with the VIM Driver (cc:Mail)

Data Type:
String

DisplayErrors

Description:
This property sets and returns the error handling state.
When setting the DisplayErrors property, the Session Control will Display a message box when an error occurs.

Usage:
b_DisplayErrors=[*form*].*MSess.***DisplayErrors**
[*form*].*MSess.***DisplayErrors**= *b_DisplayErrors*

Data Type:
BOOL

DownLoadMsg

Description:
Set this property to force a download of all new messages from the mail server to a user's Inbox during the sign-in process. Use this flag so that an application can deal with the user's complete set of messages when it signs in. When set, a progress indicator popup is displayed and automatically removed when the process is complete. Use of this flag may increase processing time.

Usage:
b_DownLoadMsg=[*form*].*MSess.***DownLoadMsg**
[*form*].*MSess.***DownLoadMsg**= *b_DownLoadMsg*

Data Type:
BOOL

ErrorText

Description:

This property sets and returns the String associated with the last error. The ErrorText is the string that appears in the Error Message Box

Usage:

sz_ErrorText$=[*form*].*MSess.***ErrorText**

Data Type:

String

ErrorNum

Description:
This property returns the Number of the Last error.

Values:

Error Code	Error Num	Description
ERROR_LOADING_LIBRARY	1	Mail Library (VIM or MAPI) could not be loaded
ERROR_NOT_CONNECTED	2	Mail X Could not complete the Logon process. Check The User and Password Properties
ERROR_EX_FUNCTION	3	Mail X Function error or Not supported by Driver

Usage:
i_ErrorNum=[*form*].*MSess.***ErrorNum**

Data Type:
Integer

FolderCount

Description:
This property returns the container category count available in your VIM Session.

Usage:
i_FolderCount=[*form*].*MSess.***FolderCount**

Remarks:
This New Property is only available for VIM Driver (cc:Mail / Notes)

Data Type:
Long Integer

FolderNum

Description:
This property sets and returns the current category number being used by the MailX Session control.

Usage:
i_FolderNum=[*form*].*MSess.***FolderNum**
[*form*].*MSess.***FolderNum** = *i_NewFolderNum*

Remarks:
This New Property is only available for VIM Driver (cc:Mail / Notes). Set FolderNum=0 when using the Inbox.

Data Type:
Long Integer

LeaveMailOnServer

Description:
The LeaveMailOnServer property sets & returns the status of your POP3 delete command.

Usage:
b_LeaveMail=[*form*].*MSess.* **LeaveMailOnServer**
[*form*].*MSess.* **LeaveMailOnServer** = *b_LeaveMail*

Remarks:
Set this property to FALSE to remove All messages from your POP3 server after downloading messages to your Local Internet Message Store. Set this property to TRUE to leave all messages in your POP3 mailbox.

Data Type:
BOOL

Logon

Example

Description:

The Logon property returns the status of your mail connection.
The Logon property is also set to establish or cancel a connection to your mail server.

Values:

Value		Description
TRUE	-1	Connection is established with your Mail server
FALSE	0	Cancel a connection previously established.

Usage:

b_LogonStatus=[*form*].*MSess.***Logon**
[*form*].*MSess.***Logon**= *b_Logon*

Data Type:

BOOL

LogonUI

Example

Description:

Set this property if the control should display a dialog box to prompt for name and password (if required). When this flag is not set, the Session Control does not display a sign-on dialog box and returns an error if the user is not signed in.

Usage:

b_LogonUI =[*form*].*MSess.***LogonUI**
[*form*].*MSess.***LogonUI**= *b_LogonUI*

Data Type:

BOOL

Mail_Type

Description:
Sets and returns the current Mail system being used by the MailX session control.

Usage:
i_MailType =[*form*].*MSess.***Mail_Type**
[*form*].*MSess.***Mail_Type**= *i_MailType*

Settings:

Value		Description
NONE	0	none mail system is currently selected. This value is useful when selecting the Mail System at Runtime.
EXCHANGE	1	Microsoft Mail or Exchange (MAPI.DLL is required)
VIM	2	cc:Mail or Notes for Windows (VIM.DLL library is required)
POP3SMTP	3	POP3/SMTP protocols.
CUSTOM	4	Custom Simple Mapi Library (see MapiCustomLibName)

Data Type:
Integer

MapiCustomLibName

Description:
Set this property if you want to establish a session with a Custom Simple Mapi Library.

Usage:
sz_MapiCustomLib$=[*form*].*MSess.* **MapiCustomLibName**
[*form*].*MSess.* **MapiCustomLibName** = *sz_MapiCustomLib$*

Data Type:
String

NewSession

Description:
Set this property if you want to establish a session other than the current one. For instance, if a mail client is already running, another MAPI electronic mail client can piggyback on the session created by the mail client application. Do not set if you want the default session (if it still exists).

Usage:
b_NewSession =[*form*].*MSess.***NewSession**
[*form*].*MSess.NewSession*= *b_NewSession*

Data Type:
BOOL

Password

Example

Description:
This property sets and returns the password that will be used when you logon your Mail system.

Usage:
sz_UserPassword$=[*form*].*MSess.***Password**
[*form*].*MSess.***Password**= *sz_UserPassword$*

Remarks:
You must set a valid Password to logon your Mail System, otherwise, an error will be generated.

Data Type:
String

POPath

Example

Description:
This property returns the Post Office path used by the current session.

Usage:
sz_UserPassword$=[*form*].*MSess.***POPath**

Remarks:
You must have an active Mail session with your server. This new property is Only available with the VIM & SMTP Driver (cc:Mail / Internet Message Store).

Data Type:
String

POPServer

Description:
This property sets and returns the current POPServer name

Usage:
sz_Server$=[*form*].*MSess.***POPServer**
[*form*].*MSess.***POPServer** =*sz_Server$*

Remarks:
This new property is Only available with the Internet Driver

Data Type:
String

SMTPServer

Description:
This property sets and returns the current SMTPServer name

Usage:
sz_Server$=[*form*].*MSess.***SMTPServer**
[*form*].*MSess.***SMTPServer** =*sz_Server$*

Remarks:
This new property is Only available with the Internet Driver

Data Type:
String

RealName

Description:
This property sets and returns the Real name of the Current User

Usage:
sz_UserName $=[*form*].*MSess.***RealName**
[*form*].*MSess.***RealName** = *sz_UserName$*

Remarks:
This new property is Only available with the Internet Driver

Data Type:
String

ReturnAddress

Description:
This property sets and returns the Return Address used when sending messages to your SMTP server.

Usage:
sz_Address$=[*form*].*MSess.* **ReturnAddress**
[*form*].*MSess.* **ReturnAddress** =*sz_Address$*

Remarks:
This new property is Only available with the Internet Driver

Data Type:
String

User

Example

Description:
This property sets and returns the name that will be use when you logon your Mail system.

Usage:
sz_UserName$=[*form*].*MSess.***User**
[*form*].*MSess.***User**= *sz_UserName$*

Remarks:
When using **Microsoft Exchange,** use this property to set the **Profile name** used when logon your Exchange Server. You must set a valid UserName to logon your Mail System, otherwise, an error will be generated.

Data Type:
String

Error Event

Description:
Occurs whenever an errors is generated by the Session Control.

Syntax:
Sub MSess_Error ()

Remarks:
If you Set the DisplayErrors property, a Message Box will be displayed before the Error Event occurs.

InvokeConfigDialog Method

Description:
This method invokes the config dialog for your current Mail Driver.

Usage:
[form].MMsg. **InvokeConfigDialog**

Remarks:
This Method is only available with the Mail eXtension OCX version. Not all Mail drivers supports this method

IsLogon Method

Example

Description:

This Method returns the current session state.

Usage:

[form].MMsg. **IsLogon**

Remarks:

Only available with the Mail eXtension OCX & DCU version. Returns TRUE for active session; otherwise return FALSE.

MailLogon Method

Description:
This Method open or close a mail session with your server

Usage:
[form].MMsg. **MailLogon** [TRUE|FALSE]

Remarks:
Only available with the Mail eXtension OCX & DCU version.

NewMessages Method

Description:
This Method check if any new message has arrived to your Container

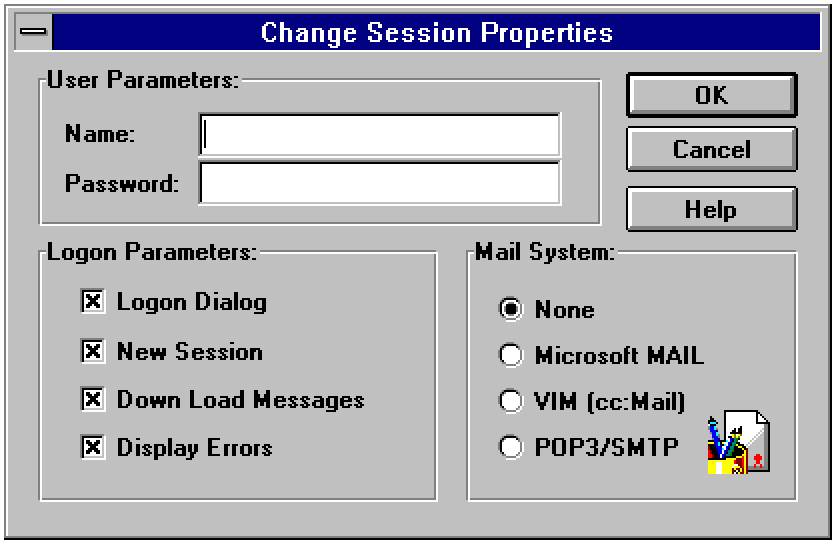
Usage:
bArrived= [form].MMsg. **NewMessages**

Remarks:
Only available with the Mail eXtension VIM Driver.

Changing Session Properties

You can change the Mail X session properties with a click on the Help Label button.

The Mail X Session Control Dialog Box is only available at Design time.



Change Session Properties

User Parameters:

Name:

Password:

Logon Parameters:

[✓ Logon Dialog]

[✓ New Session]

[✓ Down Load Messages]

[✓ Display Errors]

Mail System:

[✓ None]

Microsoft MAIL

VIM (cc:Mail)

POP3/SMTP

Session Control Dialog Box

Properties:

- About
- Action
- ChangePassword
- CurrentFolder
- CurrentUser
- DefaultPath
- DisplayErrors
- DownLoadMsg
- ErrorText
- ErrorNum
- FolderCount
- FolderNum
- Height
- Index
- Left
- LeaveMailOnServer
- Logon
- LogonUI
- Mail_Type
- MapiCustomLibName
- Name
- NewSession
- ObjRef
- Password
- POPath
- POPServer
- RealName
- ReturnAddress
- SMTPServer
- Top
- User
- Width

Events:

- Error

Methods:

- InvokeConfigDialog
- IsLogon
- MailLogon
- NewMessages

Example

The following code uses a MailX Session control, edit box and a button on a form. The code reads the user default path and try to Sign On the User in your Lotus cc:Mail Post Office:

Visual Basic Code:

```
Sub Command1_Click ()
    MSess1.DefaultPath = Text1.Text
    MSess1.LogonUI = True
    MSess1.Logon = True
    If MSess1.Logon = True Then
        MsgBox "Successful Sign On!!!"
    Else
        MsgBox "Unable to Sign On your Post Office"
    End If
End Sub
```

Delphi Code:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
     MXSession1.DefaultPath:=Edit1.Text;
     MXSession1.LogonUI:=true;
     MXSession1.Logon:=true;
     if MXSession1.Logon=true then
        Application.MessageBox('Succesful Sign On',
                               'Mail eXtension',
                               MB_OK)
     else
        Application.MessageBox('Unable to Sign on your Post Office',
                               'Mail eXtension',
                               MB_OK);
end;
```

Example

The following code uses a MailX Session control, edit box and a button on a form. The code set a new password for the current Session in your Lotus cc:Mail Post Office:

Visual Basic Code:

```
Sub Command1_Click ()
    MSess1.ChangePassword = Text1.Text
    If MSess1.ErrorNum = 0 Then
        MsgBox "Password Change!!!"
    Else
        MsgBox "Unable to Change Password"
    End If
End Sub
```

Delphi Code:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
     MXSession1.ChangePassword:=Edit1.Text;
     if MXSession1.ErrorNum=0 then
        Application.MessageBox('Password Changed',
                               'Mail eXtension',
                               MB_OK)
     else
        Application.MessageBox('Unable to change Password',
                               'Mail eXtension',
                               MB_OK);
end;
```

Example

The following code uses a MailX Session control, MailX Message Control, MailX Form Control, edit box and a button on a form. The code set the Current Folder or category indicated by the user and returns the number of Message availables in your Folder:

Visual Basic Code:

```
Sub Command1_Click ()
    MSess1.CurrentFolder = Text1.Text
    NumMsg = MMsg1.MsgCount
    MsgBox "Num Messages=" + Str$(NumMsg)
End Sub
```

Delphi Code:

```
procedure TForm1.Button1Click(Sender: TObject);
var
   szNumMsg: string;
begin
     MXSession1.CurrentFolder:=Edit1.Text;
     szNumMsg:='Num Messages='+IntToStr(MXMessage1.MsgCount)+#0;
     Application.MessageBox(@szNumMsg[1],
                            'Mail eXtension',
                            MB_OK)
end;
```

Example

The following code uses a MailX Session control, 2 Edit Boxes and a button on a form. The code Sign On the User indicated in the Edit Box.

Visual Basic Code:

```
Sub Command1_Click ()
    MSess1.User = Text1.Text
    MSess1.Password = Text2.Text
    MSess1.Logon = True
    If MSess1.Logon = True Then
        MsgBox "Succesful Sign On"
    Else
        MsgBox "Unable to Sign On"
    End If
End Sub
```

Delphi Code:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
     MXSession1.User:=Edit1.Text;
     MXSession1.Password:=Edit2.Text;
     MXSession1.Logon:=true;
     if MXSession1.Logon=true then
        Application.MessageBox('Successful Sign On',
                               'Mail eXtension',
                               MB_OK)
     else
        Application.MessageBox('Unable to Sign On',
                               'Mail eXtension',
                               MB_OK);

end;
```

Example

The following code uses a MailX Session control, 2 Edit Boxes and a button on a form. The code Sign On the User indicated in the Edit Box.

Visual Basic Code:

```
Sub Command1_Click ()
    MSess1.User = Text1.Text
    MSess1.Password = Text2.Text
    MSess1.Logon = True
    If MSess1.IsLogon Then
        MsgBox "Succesful Sign On"
    Else
        MsgBox "Unable to Sign On"
    End If
End Sub
```

Delphi Code:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
     MXSession1.User:=Edit1.Text;
     MXSession1.Password:=Edit2.Text;
     MXSession1.Logon:=true;
     if MXSession1.IsLogon then
        Application.MessageBox('Successful Sign On',
                               'Mail eXtension',
                               MB_OK)
     else
        Application.MessageBox('Unable to Sign On',
                               'Mail eXtension',
                               MB_OK);

end;
```

Example

The following code uses a MailX Session control, 2 Edit Boxes and a button on a form. The code Sign On the User indicated in the Edit Box and return The Current Lotus cc:Mail Post Office Path.

Visual Basic Code:

```
Sub Command1_Click ()
    MSess1.User = Text1.Text
    MSess1.Password = Text2.Text
    MSess1.Logon = True
    If MSess1.IsLogon Then
        MsgBox PostOffice Path=+Msess1.POPath
    Else
        MsgBox "Unable to Sign On"
    End If
End Sub
```

Delphi Code:

```
procedure TForm1.Button1Click(Sender: TObject);
var
   szOffice: string;
begin
     MXSession1.User:=Edit1.Text;
     MXSession1.Password:=Edit2.Text;
     MXSession1.LogonUI:=true;
     MXSession1.Logon:=true;
     if MXSession1.IsLogon then
     begin
        szOffice:='PostOffice Path='+MXSession1.POPath+#0;
        Application.MessageBox(@szOffice[1],
                               'Mail eXtension',
                               MB_OK);
     end
     else
        Application.MessageBox('Unable to Sign On',
                               'Mail eXtension',
                               MB_OK);

end;
```

Example

The following code uses a MailX Session control, 2 Edit Boxes and a button on a form. The code Sign On the User indicated in the Edit Box. Mail eXtension will display the Sign On Dialog Box if user activate The LogonUI property

Visual Basic Code:

```
Sub Command1_Click ()
    MSess1.User = Text1.Text
    MSess1.Password = Text2.Text
    If MsgBox("Try to Display Sign On?", 4, "Mail eXtension") = 6 Then
        MSess1.LogonUI = True
    Else
        MSess1.LogonUI = False
    End If

    MSess1.Logon = True
    If MSess1.Logon = True Then
        MsgBox "Successful Sign On!!"
    Else
        MsgBox "Unable to Sign On"
    End If
End Sub
```

Delphi Code:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
     MXSession1.User:=Edit1.Text;
     MXSession1.Password:=Edit2.Text;
     if MessageBox(0,'Try to display Sign On Dialog','Mail eXtension',4)=6 then
        MXSession1.LogonUI:=true
     else
        MXSession1.LogonUI:=False;
     MXSession1.Logon:=true;
     if MXSession1.IsLogon then
     begin
        Application.MessageBox('Successful Sign On',
                               'Mail eXtension',
                               MB_OK);
     end
     else
        Application.MessageBox('Unable to Sign On',
                               'Mail eXtension',
                               MB_OK);
end;
```

Mail X Message Control

Properties Methods Events

Overview:

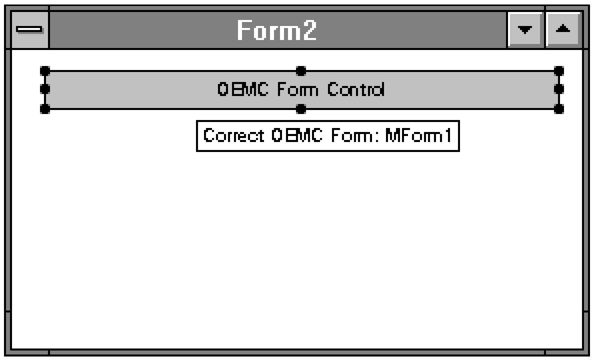
The Mail X Message Control is used to handle any message in your Inbox. You can also compose (New, Reply, Forward) a new message to be sent through your Mail Server.

Because Mail X Message Control handle one Inbox message, you must add many controls as you need in your Visual Basic Form to handle different messages simultaneously.

The Message Control only handle information relative to the E-Mail Message (Subject, Note Text, etc.). To obtain or set the File Attachments or the Message Recipients you should use the appropriate Mail X controls.

Click your Right button and change the Message Control properties with a Mail X Message Control DialogBox

Note: You must have a Mail X Form Control before adding Message Control on your Visual Basic Form



Action

Example Example2

Description:
This property set a new Action in your Mail X Message Custom Control.

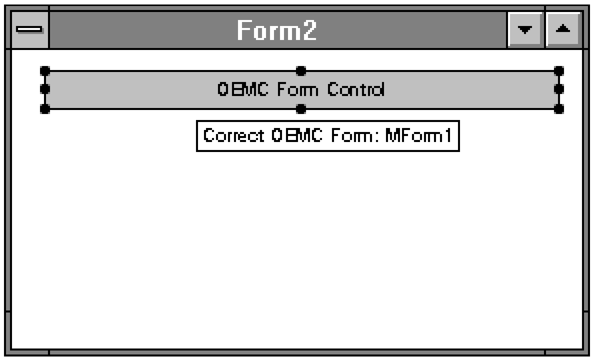
Usage:
[form].MMsg. **Action**=i_NewAction

Values:

Action ID	Number	Description
ACTION_DELETEMAIL	1	Delete the current message from the message inbox
ACTION_FINDNEXT	2	Fetch the next message of the message inbox.
ACTION_FINDFIRST	3	Fetch the first message of the message inbox
ACTION_CLEARMSG	4	Clear the current message
ACTION_CLEARRECIP	5	Clear the message recipient descriptors
ACTION_CLEARFILE	6	Clear the file attachment descriptors of the current message
ACTION_NEW	7	Composes a message. Clears all of the components of the compose buffer
ACTION_REPLY	8	Replies to a message. Copies the currently fetched message to the compose buffer as a reply and adds RE: to the beginning of the Subject line.The currently indexed message originator becomes the outgoing message recipient, then text is copied, and so on.
ACTION_FORWARD	9	Forwards a message. Copies the currently indexed message to the compose buffer as a forwarded message and adds FW: to the beginning of the Subject line.
ACTION_REPLYALL	10	Replies to all message recipients. Same as Reply, except that all other To: and CC: recipients are maintained.
ACTION_COPYMSG	11	Copy the Subject and the NoteText from the inbox message to the compose buffer.
ACTION_SAVEMSG	12	Save the current message in the inbox
ACTION_SENDMSG	13	The current message is sent. If the DisplaySendDialog property is set, all message properties associated with a message being built in the compose buffer form the basis for the displayed message dialog box. Changes made in the dialog box, however, do not alter information in the compose buffer.

Remarks:
This Property is runtime write only. An Error event will be generated if the action fails.

Data Type:
Integer



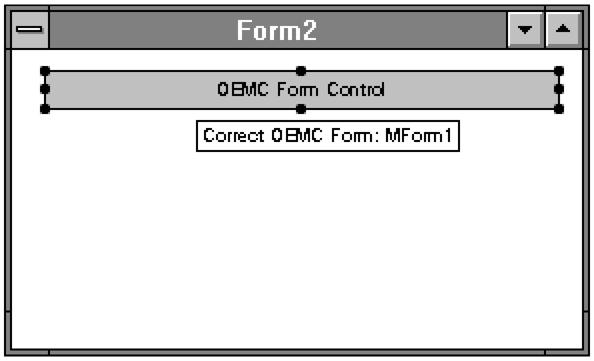
BindString

Description:
Returns the binding connection string

Usage:
hsz_BindingString$=[form].MMsg. **BindString**

Remarks:
The string contains the Form and the name of the bound control

Data Type:
String



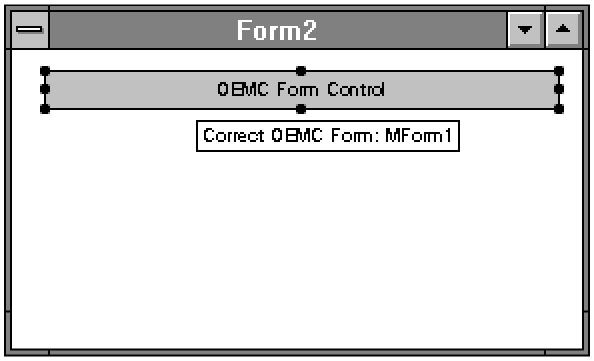
BodyAsFile

Example

Description:
Set this property when you want the message body written to a temporary file and added to the attachment list as the first attachment.

Usage:
b_BodyAsFile=[form].MMsg. **BodyAsFile**
[form].MMsg. **BodyAsFile**=*b_BodyAsFile*

Data Type:
BOOL

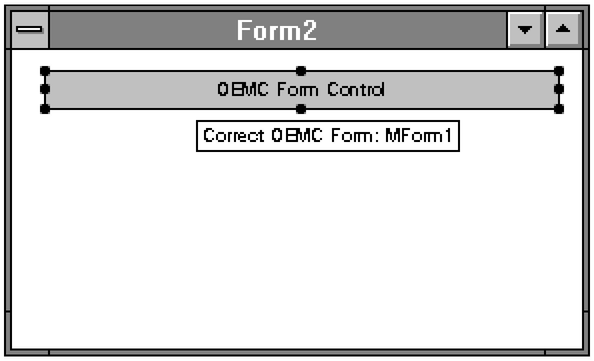


ConversationID

Description:
Returns a string indicating the conversation thread ID to which this message belongs.

Usage:
hsz_ConversationID$=[form].MMsg. **ConversationID**

Data Type:
String



DisplayErrors

Description:

This property sets and returns the error handling state.
When setting the DisplayErrors property, the Message Control will Display a message box when an error occurs.

Usage:

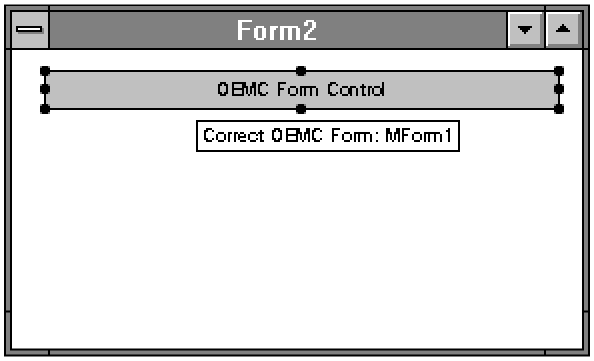
*b_DisplayErrors=[form].MMsg.***DisplayErrors**
[form].MMsg. **DisplayErrors**=*b_DisplayErrors*

Remarks:

Regardless of this value, the ErrorNum and ErrorText properties are changed before an Error Event is generated

Data Type:

BOOL



DisplaySendDialog

Example

Description:

Set this property if the message control should display a dialog box to prompt for recipients and other sending options. When this flag is not set, the message control does not display a dialog box, but at least one recipient must be specified.

Usage:

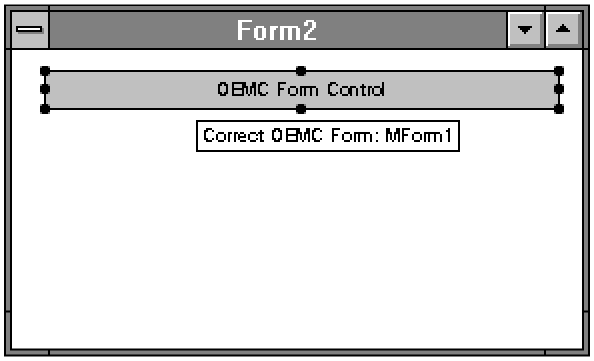
b_DisplayDialog=[form].MMsg. **DisplaySendDialog**
[form].MMsg. **DisplaySendDialog**=*b_DisplayDialog*

Remarks:

See also Action property.

Data Type:

BOOL



EnvelopeOnly

Example

Description:

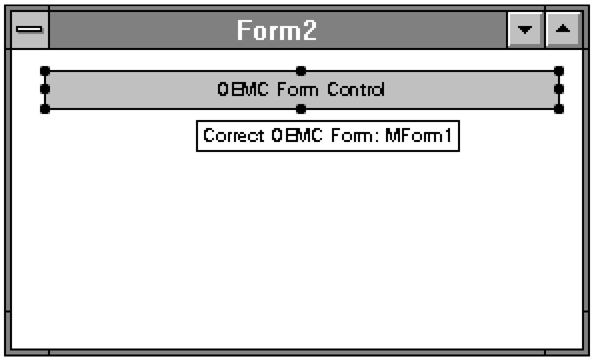
Set this property when you don't want the function to copy file attachments to temporary files or return the note text. All other message information (except for temporary filenames) is returned. Setting this flag usually reduces the processing time required for the function.

Usage:

b_EnvelopeOnly =[form].MMsg. **EnvelopeOnly**
[form].MMsg. **EnvelopeOnly**=*b_EnvelopeOnly*

Data Type:

BOOL



ErrorNum

Example

Description:
This property returns the number of the last error.

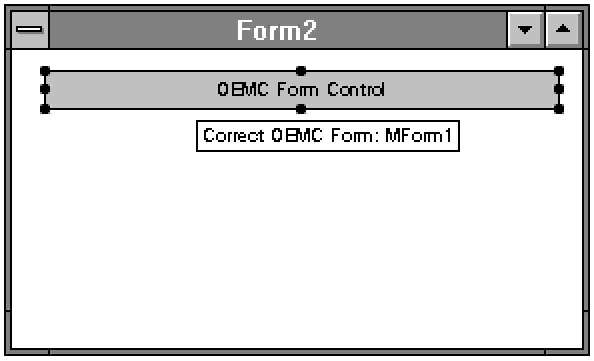
Usage:
i_ErrorNum%=[form].MMsg. **ErrorNum**

Values:

Error Code	Number	Description
ERROR_MSG_NOTCONNECTED	1	Session has not been established with your mail server.
ERROR_MSG_NOTFECTH	2	Unable to execute the operation because the message Control is empty
ERROR_MSG_NOT_BOUND	3	The Message Control is NOT bound
ERROR_MSG_SENDMAIL	4	Unable to Send Current Message
ERROR_MSG_SAVEMSG	5	Unable to save the current message
ERROR_MSG_EMPTYMSG	6	The current message is Empty
ERROR_MSG_DELETE	7	Unable to delete the current message
ERROR_MSG_SETMSGID	8	Unable to find the message by the Msg ID
ERROR_MSG_FINDNEXT	9	Unable to fetch the next message
ERROR_MSG_FINDFIRST	10	Unable to fetch the first message
ERROR_MSG_FETCHMSG	11	Unable to fetch the inbox message
ERROR_MSG_EX_FUNCTION	13	Extended function Error or NOT supported by Mail eXtension Driver
ERROR_MSG_SEND_BADRECIP	14	Unable to Send Current Message. (BAD RECIPIENT ERROR)

Remarks:
This property is set to zero (0) before another action gets executed.

Data Type:
Integer



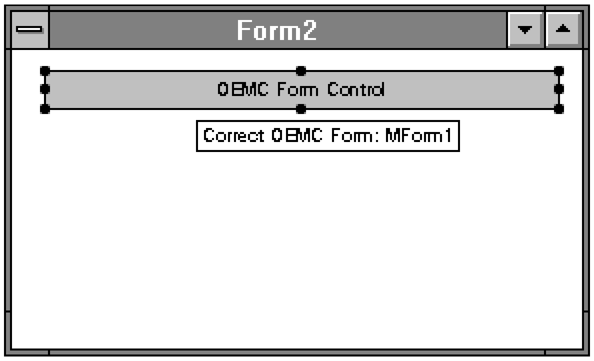
ErrorText

Description:
This properties contains the last error text.

Usage:
hsz_ErrorText$=[form].MMsg. **ErrorText**

Remarks:
This property is set regardless of the DisplayErrors property.

Data Type:
String



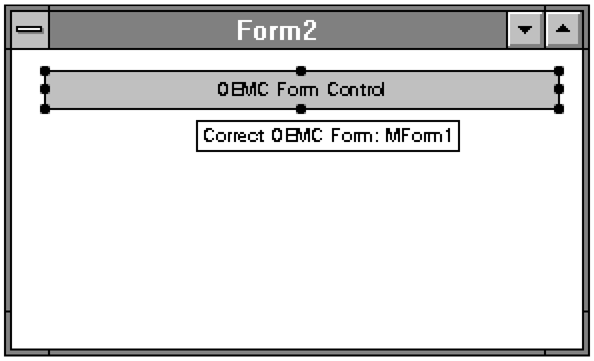
FastFetch

Description:
Use this property to execute a **partial fetch**. *FastFetch* is useful when scanning inbox, If Mail eXtension Driver supports Fast Fetch you will notice 100% faster performance

Usage:
bFast=[form].MMsg. **FastFetch**
[form].MMsg. **FastFetch** = *bFast*

Remarks:
When using Fast Fetch, Mail eXtension Driver will only return: MsgID, Subject, Originator, Date, IsUnread, MsgType & ReturnReceipt Properties.

Data Type:
Boolean



FetchMsg

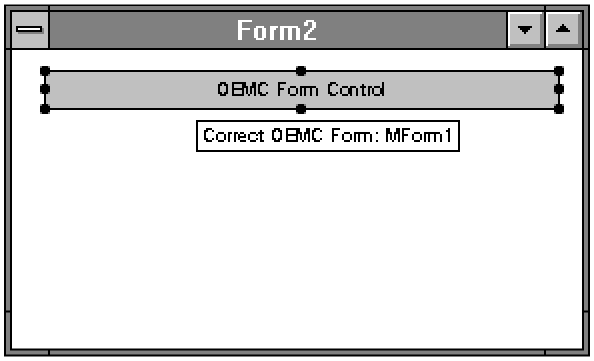
Example

Description:
Use this property to Fetch an inbox message

Usage:
i_MsgNum=[form].MMsg. **FetchMsg**
[form].MMsg. **FetchMsg**=*i_MsgNum*

Remarks:
The property will fetch the message number specified. This property returns zero (0) if no message is fetched.

Data Type:
Long Integer

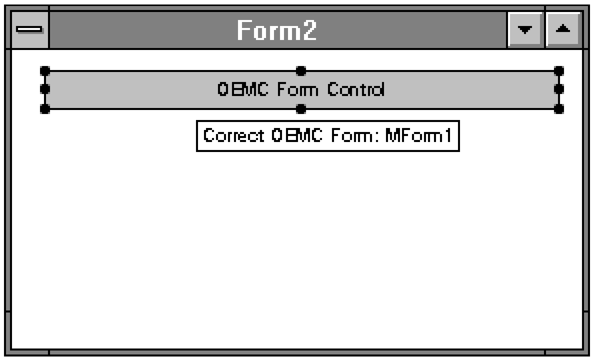


FetchMsgType

Description:
Sets and returns the message type. To specify an interpersonal mail message, specify an empty string, "".

Usage:
b_FetchMsgType $=*[form].MMsg.* **FetchMsgType**
[form].MMsg. **FetchMsgType**=*b_FetchMsgType$*

Data Type:
String



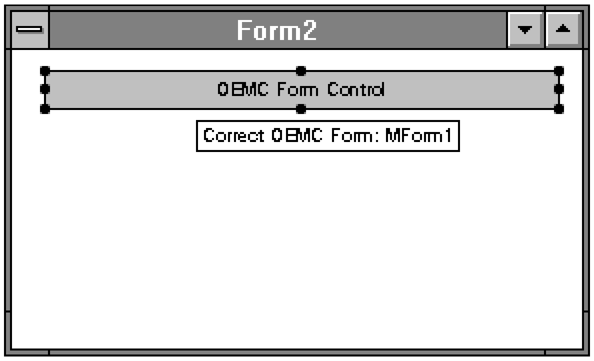
IsBinded

Description:
Returns the binding state of the Mail X Message Control.

Usage:
b_IsBinded=[form].MMsg. **IsBinded**

Remarks:
Returns TRUE if the control is succesfully bound, otherwise a FALSE is returned

Data Type:
BOOL



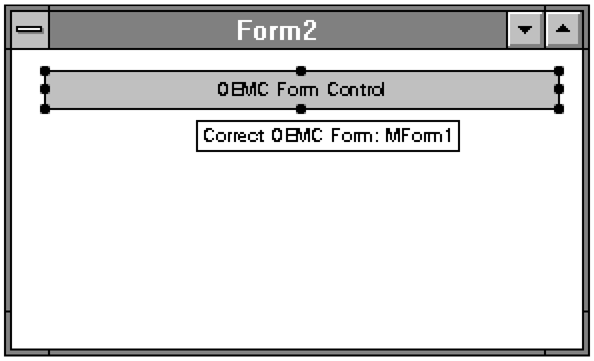
MarkAsRead

Examples

Description:
Set MarkAsRead when you want the Mail X Message Control to mark the message as read. Any unsuccessful fetch leaves the message unread.

Usage:
b_MarkAsRead =[form].MMsg. **MarkAsRead**
[form].MMsg. **MarkAsRead**= *b_MarkAsRead*

Data Type:
BOOL



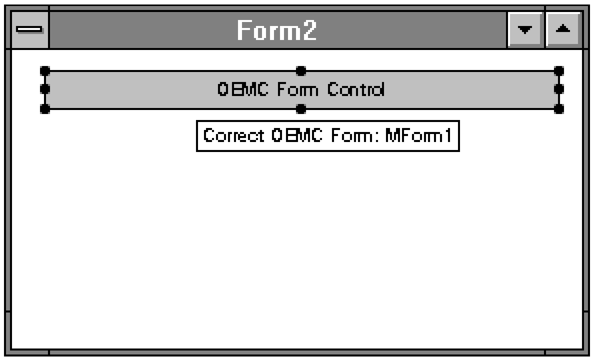
MsgCount

Example

Description:
Returns the number of Messages available in the Inbox.

Usage:
i_MessagetCount%=[form].MMsg. **MsgCount**

Data Type:
Long Integer



MsgID

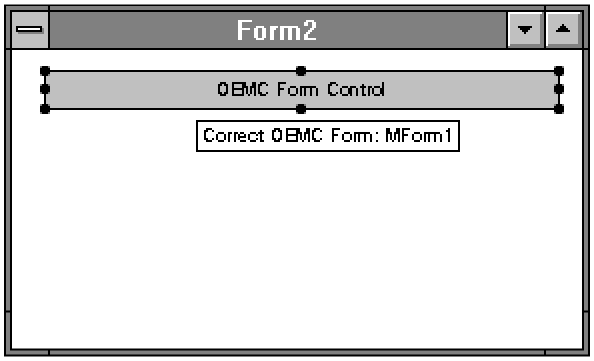
Example

Description:
Returns the MessageID of the current message.

Usage:
hsz_MsgID$=[form].MMsg. **MsgID**
[form].MMsg. **MsgID**= *hsz_MsgID$*

Remarks:
When SETTING this property, the Mail X Message Control will search for an inbox box message that match the specified message ID; if found, the message is fetched, otherwise the Mail X Message control is flushed.

Data Type:
String



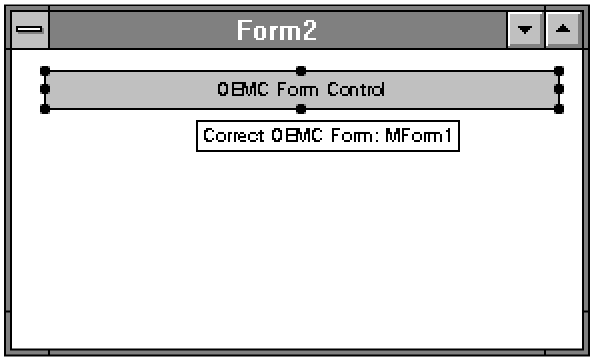
MsgType

Description:
Returns the message type of the current message.

Usage:
hsz_MsgType$=[form].MMsg. **MsgType**
[form].MMsg. **MsgType**= *hsz_MsgType$*

Remarks:
A empty string indicates an interpersonal message (IPM) type.

Data Type:
String



NoteText

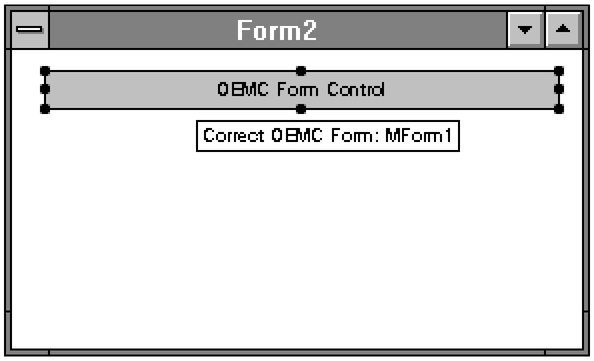
Example

Description:
Sets an returns the note text of the current message.

Usage:
hsz_NoteText$=[form].MMsg. **NoteText**
[form].MMsg. **NoteText**= *hsz_NoteText$*

Remarks:
An Empty string is returned if the Message control is flushed or empty.

Data Type:
VB: String
Delphi: TStrings



Priority

Description:
Sets an returns the priority of the current message.

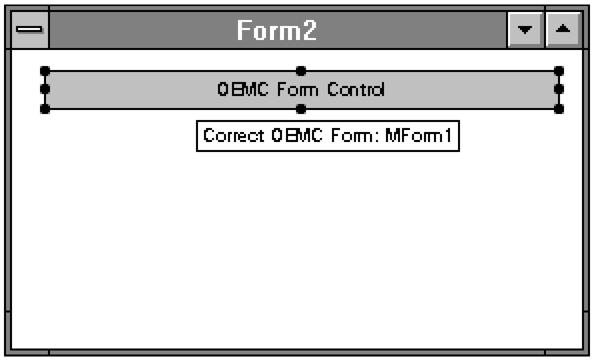
Usage:
i_Priority=[form].MMsg. **Priority**
[form].MMsg. **Priority** = *i_Priority*

Settings:

ID	Value	Description
NORMAIL	0	Normal Priority
LOW	1	Low importance
HIGH	2	High importance

Remarks:
Only available for VIM Driver

Data Type:
Integer

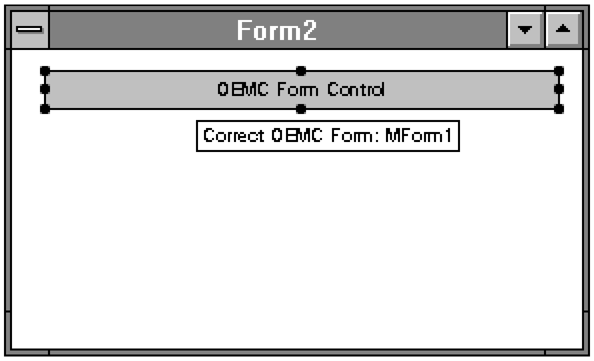


ReceiptRequested

Description:
Sets and returns the state of the receipt requested flag.

Usage:
b_ReceiptRequested =[form].MMsg. **ReceiptRequested**
[form].MMsg. **ReceiptRequested**= *b_ReceiptRequested*

Data Type:
BOOL



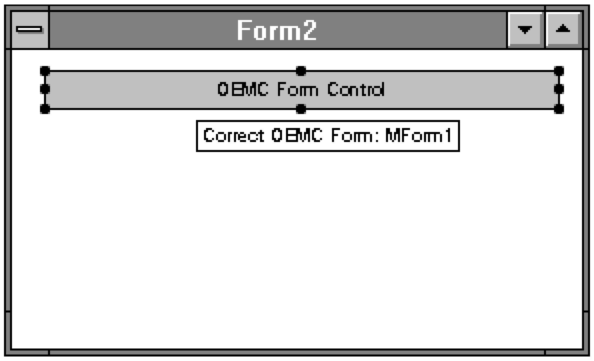
SaveMsg

Description:
Sets and returns the state of the Save message flags

Usage:
b_ SaveMsg =[form].MMsg. **SaveMsg**
[form].MMsg. **SaveMsg** = *b_ SaveMsg*

Remarks:
Only available for VIM Driver

Data Type:
BOOL

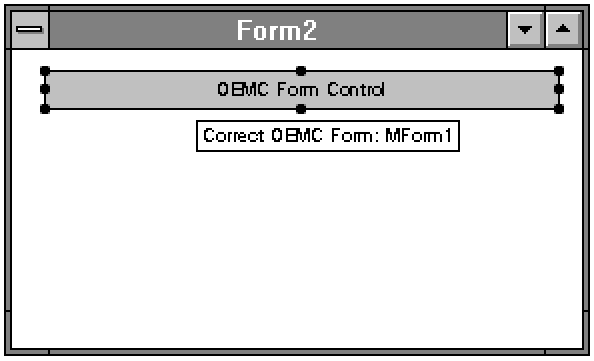


SentMsg

Description:
Sets and returns the state of the Sent message flags

Usage:
b_ SentMsg =[form].MMsg. **SentMsg**
[form].MMsg. **SentMsg** = *b_ SentMsg*

Data Type:
BOOL

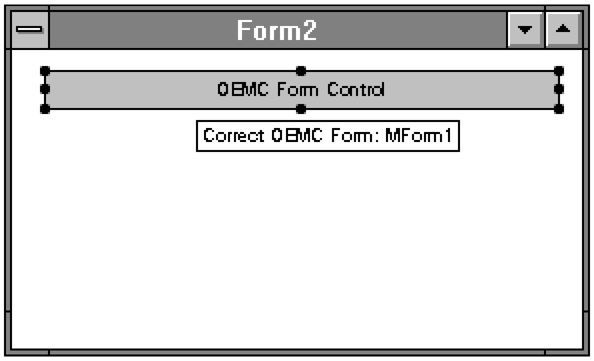


SortMsg

Description:
Set this property if you want to sort the inbox messages in the same order as received. Fetching a message may take longer if this flag is set.

Usage:
b_SortMsg =[form].MMsg. **SortMsg**
[form].MMsg. **SortMsg**= *b_SortMsg*

Data Type:
BOOL



Subject

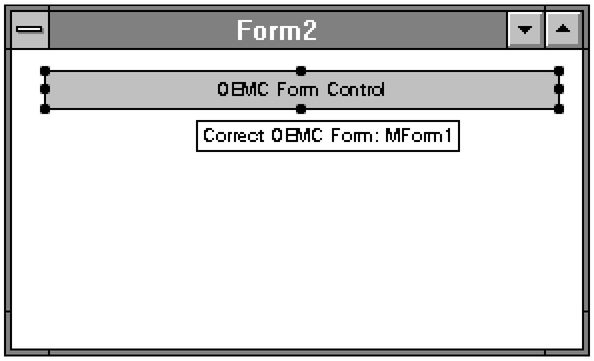
Example

Description:
Sets an returns the subject   of the current message.

Usage:
hsz_Subject$=[form].MMsg. **Subject**
[form].MMsg. **Subject** = *hsz_Subject$*

Remarks:
An Empty string is returned if the Message control is flushed or empty.

Data Type:
String



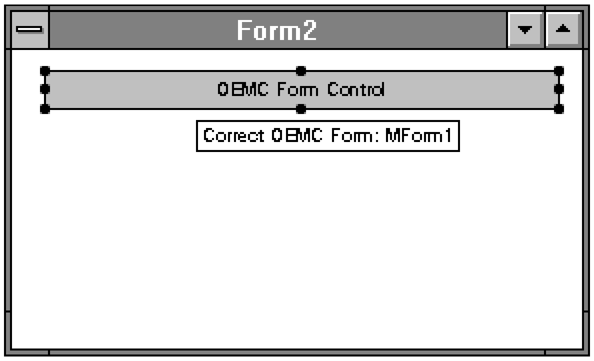
SuppressAttach

Description:
Set this property when you don't want Message Control to copy file attachments, but instead just return message text.

Usage:
b_SuppressAttach =[form].MMsg. **SuppressAttach**
[form].MMsg. **SuppressAttach**= *b_SuppressAttach*

Remarks:
This flag is ignored if EnvelopeOnly is set. The flag should reduce the time required by the Message Control.

Data Type:
BOOL



TimeFormat

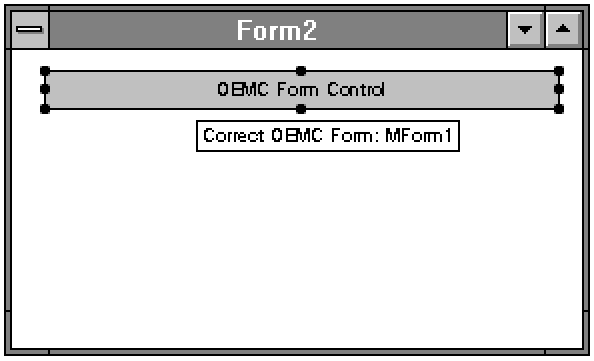
Description:
Sets and returns the time format being used when reading the TimeReceived property.

Usage:
hsz_TimeFormat$=[form].MMsg. **TimeFormat**
[form].MMsg. **TimeFormat**= *hsz_TimeFormat$*

Values:

Command	Description
%A	Full weekday name
%a	Abbreviated weekday name
%B	Full month name
%b	Abbreviated month name
%d	Day of the month as a decimal number (01-31)
%H	Hour in 24 hour format (00-23)
%I	Hour in 12 hour format (01-12)
%j	Day of the year as a decimal number (001-366)
%m	Month as a decimal number (01-12)
%M	Minute as a decimal number (00-59)
%p	Current locales AM/PM indicator for a 12-hour clock
%S	Second as a decimal number (00-59)
%w	Weekday as a decimal number (0-6)
%y	Year without the century as a decimal number (00-99)
%Y	Year with the century as a decimal number
%%	Percent sign

Data Type:
String

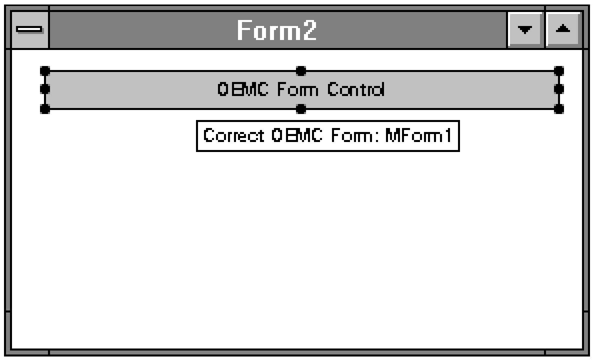


TimeReceived

Description:
Returns a string indicating the date a message is received. The format depends of the TimeFormat property.

Usage:
hsz_TimeReceived$=[form].MMsg. **TimeReceived**

Data Type:
String



UnreadMsg

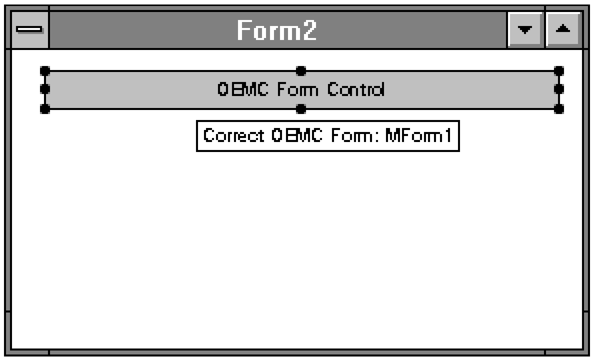
Example

Description:
Set this property if the Mail X Message Control should enumerate only unread messages.

Usage:
b_UnreadMsg =[form].MMsg. **UnreadMsg**
[form].MMsg. **UnreadMsg**= *b_UnreadMsg*

Remarks:
When this flag is not set, all messages are available.

Data Type:
BOOL



WorkingMsg

Example

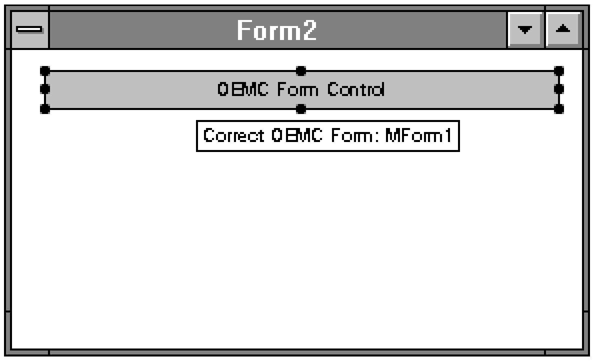
Description:
Select the current message you are working with.

Usage:
i_WorkingMsg =[form].MMsg. **WorkingMsg**
[form].MMsg. **WorkingMsg**= *i_WorkingMsg*

Settings:

ID	Value	Description
INBOX_MSG	0	Inbox Message
COMPOSE_MSG	1	Compose message

Data Type:
Integer

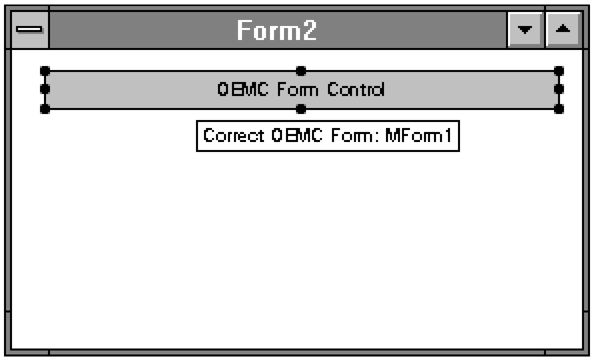


Error Event

Description:
Occurs whenever an errors is generated by the Message Control.

Syntax:
Sub MMsg_Error ()

Remarks:
If you Set the DisplayErrors property, a Message Box will be displayed before the Error Event occurs.



AddItem Method

Example

Description:

Adds an item to the Mail X Message control at run time.

Usage:

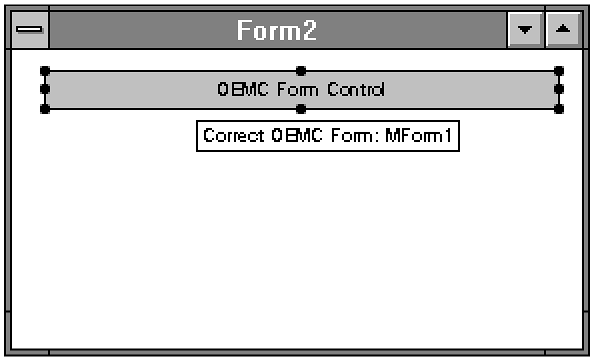
[form].MMsg. **AddItem** item[,index%]

Settings:

Value	Number	Description
NEW_MSG	0	Create a new message with a subject indicated by the item string
REPLY_MSG	1	Reply the current fetched message
FORWARD_MSG	2	Forward the current fetched message
REPLYALL_MSG	3	Reply all recipients of the current fetched message

Remarks:

if no index is specified, a new message is created.



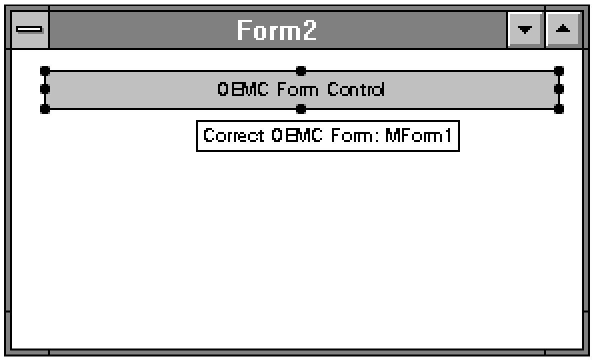
Remove Method

Description:
Remove the current message item.

Usage:
[form].MMsg. **RemoveItem** index%

Settings:

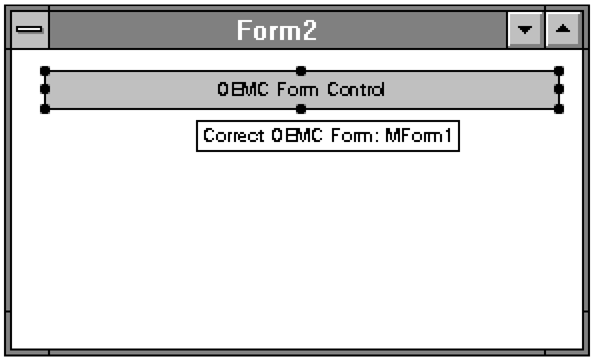
Value	Number	Description
CLEAR_MSG	0	Flush the current message
CLEAR_RECIP	1	Flush the recipients of the current message
CLEAR_FILE	2	Flush the file attachment of the current message



Clear Method

Description:
Flush the current message.

Usage:
[form].MMsg. **Clear**

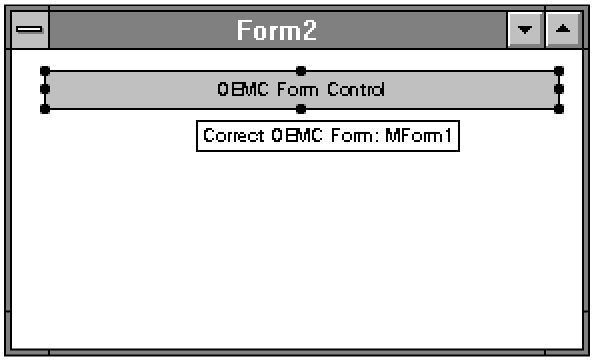


MoveToFolder Method

Description:
Move the current message.to the specified folder.

Usage:
[form].MMsg. **MoveToFolder** folderName

Remarks:
If the specified folder doesnt exist, Mail eXtension will create it in your Post office. (Only available with cc:Mail). When using the Mail eXtension VBX version, use the MoveToFolder property (*[form].MMsg.* **MoveToFolder=**folderName)

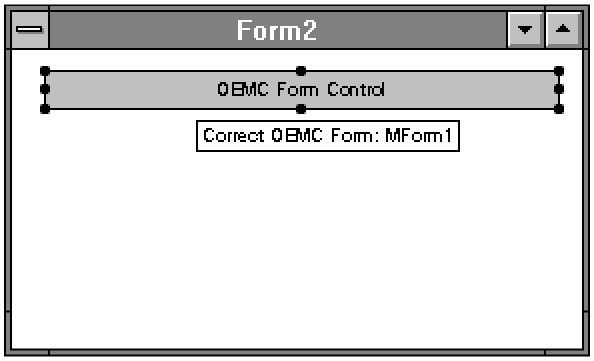


RemoveFromFolder Method

Description:
Remove the current message.from the specified folder.

Usage:
[form].MMsg. **RemoveFromFolder** folderName

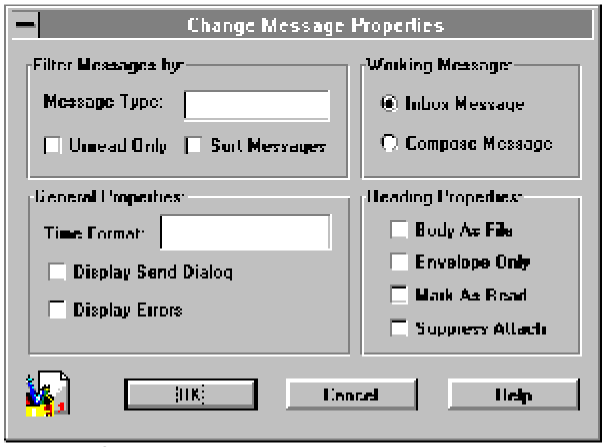
Remarks:
Only available with cc:Mail sessions. When using the Mail eXtension VBX version, use the RemoveFromFolder property (*[form].MMsg.* **RemoveFromFolder=**folderName)



Changing Message Control Properties

You can change the Mail X Message properties with a click on the Help Label button.

The Mail X Message Control Dialog Box is only available at Design time.



Message Control Dialog Box

Properties:

- About
- Action
- BindString
- BodyAsFile
- ConversationID
- DisplayErrors
- DisplaySendDialog
- EnvelopeOnly
- ErrorNum
- ErrorText
- FastFetch
- FetchMsg
- FetchMsgType
- Height
- Index
- IsBinded
- Left
- MarkAsRead
- MsgCount
- MsgID
- MsgType
- Name
- NoteText
- ObjRef
- Priority
- ReceiptRequested
- SaveMsg
- SentMsg
- SortMsg
- Subject
- SuppressAttach
- Tag
- TimeFormat
- TimeReceived
- Top
- UnreadMsg
- Width
- WorkingMsg

Events:

- Error

Methods:

- AddItem
- BindWith
- RemoveItem
- Clear
- MoveToFolder
- RemoveFromFolder

Example

The following code uses a MailX Session control, Message Control, Recipient Control, 3 Edit Boxes and a button on a form. The code compose and Send an e-mail with the Subject, NotePart and the Receipient Address indicated by the User

Visual Basic Code:

```
Sub Command2_Click ()
    If MSess1.Logon = True Then
        MReci1.Clear
        MReci1.ResolveName = szAddress.Text
        If MReci1.ResolveName = "" Then
            MsgBox "Recipient NOT FOUND"
        Else
            MReci1.Action = ACTION_ADDRECIPIENT
            MMsg1.Action = ACTION_NEW
            MMsg1.WorkingMsg = COMPOSE_MSG
            MReci1.Action = ACTION_RECIP_SET
            MMsg1.Subject = szSubject
            MMsg1.NoteText = szNoteText

            MMsg1.Action = ACTION_SENDMSG

            If MMsg1.ErrorNum = 0 Then
                MsgBox "Message has been sent"
            Else
                MsgBox "Error Sending The Mail message"
            End If
        End If
    Else
        MsgBox "No Active Session available"
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control, Edit Box and a button on a form. The code Fetch the Next Message and display the Subject in the Edit Box

Visual Basic Code:

```
Sub Command4_Click ()
    MMsg1.Action = ACTION_FINDNEXT
    Text1 = MMsg1.Subject
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control, Edit Box and a button on a form. The code Fetch the Next Message and save the Message Note Part as a file attachment in your temporary directory

Visual Basic Code:

```
Sub Command4_Click ()
    MMsg1.BodyAsFile = True
    MMsg1.Action = ACTION_FINDNEXT
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control, 3 Edit Boxes and a button on a form. The code compose and Send an e-mail with the Subject, NotePart and the Receipient Address indicated by the User. The Send Dialog is shown if the user set the DisplaySendDialog property

Visual Basic Code:

```
Sub Command2_Click ()
    If MSess1.Logon = True Then
        MReci1.Clear
        MReci1.ResolveName = szAddress.Text
        If MReci1.ResolveName = "" Then
            MsgBox "Recipient NOT FOUND"
        Else
            MReci1.Action = ACTION_ADDRECIPIENT
            MMsg1.Action = ACTION_NEW
            MMsg1.WorkingMsg = COMPOSE_MSG
            MReci1.Action = ACTION_RECIP_SET
            MMsg1.Subject = szSubject
            MMsg1.NoteText = szNoteText

            If MsgBox("Display Send Dialog", 4, "Mail eXtension") = 6 Then
                MMsg1.DisplaySendDialog = True
            Else
                MMsg1.DisplaySendDialog = False
            End If
            MMsg1.Action = ACTION_SENDMSG


            If MMsg1.ErrorNum = 0 Then
                MsgBox "Message has been sent"
            Else
                MsgBox "Error Sending The Mail message"
            End If
        End If
    Else
        MsgBox "No Active Session available"
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control, 2 Edit Boxes and a button on a form. The code Fetch the Message number indicated by the User and display the Subject in the Edit Box if the Fetch operation is successful

Visual Basic Code:

```
Sub Command3_Click ()
    nNum = Val(Text1.Text)
    MMsg1.FetchMsg = nNum
    If MMsg1.ErrorNum = 0 Then
        Text2.Text = MMsg1.Subject
    Else
        MsgBox "Error FetchMsg"
    End If

End Sub
```

Example

The following code uses a MailX Session control, Message Control, Edit Box and a button on a form. The code counts the number of message availables in your Inbox or current Folder.

Visual Basic Code:

```
Sub Command1_Click ()
    Text1.Text = Str$(MMsg1.MsgCount)
End Sub
```

Example

The following code uses a MailX Session control, 2 Message Controls, Edit Box and a button on a form. The code Read the Message ID from one message control and display this value in the Edit Box. The Second Message Control will fetch the same Message if the User Indicate this action.

NOTE: You can also use the = operator to clone a Message Control.

Visual Basic Code:

```
Sub Command1_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    Text1.Text = MMsg1.MsgID
    If MsgBox("Copy this Message to another Control?", 4, "Mail eXtension") = 6 Then
        MMsg2.MsgID = MMsg1.MsgID
        Text2.Text = MMsg2.MsgID
    End If
End Sub
```

Clone Example

```
Sub Command1_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    Text1.Text = MMsg1.MsgID
    If MsgBox("Copy this Message to another Control?", 4, "Mail eXtension") = 6 Then
        MMsg2 =.MsgID
     End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Edit Box and a button on a form. The code Fetch the Next Message and return the Message Subject.

Visual Basic Code:

```
Sub Command4_Click ()
    MMsg1.EnvelopeOnly = True
    MMsg1.Action = ACTION_FINDNEXT
    Text1.Text = MMsg1.Subject
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Edit Box and a button on a form. The code Fetch the Next Message and return the Message Subject. If the Message is Unread, MAILX Message Control will mark it as Read when fetching the new message.

Visual Basic Code:

```
Sub Command4_Click ()
    MMsg1.MarkAsRead = True
    MMsg1.Action = ACTION_FINDNEXT
    Text1.Text = MMsg1.Subject
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Edit Box and a button on a form. The code Fetch the Next Message and return the Message Note Part.

Visual Basic Code:

```
Sub Command4_Click ()
      MMsg1.EnvelopeOnly = False
      MMsg1.Action = ACTION_FINDNEXT
      Text1.Text = MMsg1.NoteText
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Edit Box and a button on a form. The code compose a new message a set the Subject Indicated by the User

Visual Basic Code:

```
Sub Command4_Click ()
            MMsg1.Action = ACTION_NEW
            MMsg1.WorkingMsg = COMPOSE_MSG
            MMsg1.Subject = Text1.Text
End Sub
```

Example

The following code uses a MailX Session control, Message Control and a button on a form. The code Fetch the First Message and Reply the message

Visual Basic Code:

```
Sub Command4_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    If MMsg1.ErrorNum = 0 Then
        MMsg1.AddItem "", 1
        MMsg1.Action = ACTION_SENDMSG
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control and a button on a form. The code reads all the Unread Messages availables in the Current Folder or Inbox

Visual Basic Code:

```
Sub Command2_Click ()
    If MSess1.Logon = True Then
        MMsg1.UnreadOnly = True
        MMsg1.MarkAsRead = True
        MMsg1.Action = 3
        While MMsg1.ErrorNum = 0
            ' You Should Process your Message
            MsgBox MMsg1.Subject, 0, "Subject"
            DoEvents
            MMsg1.Action = 2
        Wend
        MsgBox "Unread Scanning Complete!!!"
    End If
End Sub
```

Recipient Control

Properties Methods Events

Overview:

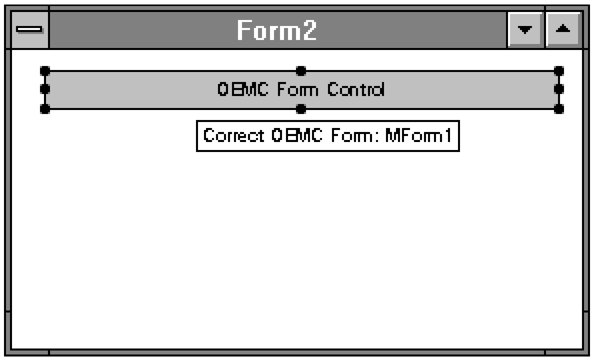
The Mail X Recipient Control is used to handle the recipients associated with a message contained in a Mail X Message Control. You can query or add new recipients to the current message contained in the Message Control.

The Recipient Control only handle information relative to the recipients of a message (*Name, address, class, etc*). To obtain or set the recipients of a mail message, BIND the Mail X Recipient Control to the Message Control.

Mail eXtension *NOW* supports Custom Address. You can send you message to any Recipient address you specify. Mail eXtension Custom Address can be used to deliver any message to different gateway defined in your PostOffice or Mail Server.

Click your Right button and change the Recipient Control properties with a Mail X Recipient Control DialogBox

Note: You must have a Mail X Form Control before
adding Mail X Recipient Control on your Visual Basic Form



Action

Example Example2

Description:
This property set a new Action in your Mail X Recipient Custom Control.

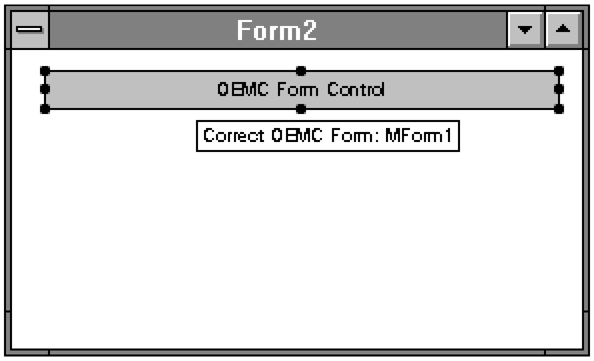
Usage:
[form].MRecipient. **Action**=i_NewAction

Values:

Action ID	Number	Description
ACTION_DETAILS	1	Display a Details Dialog box of the current Recipient
ACTION_ADDRESS	2	Display the Default ADDRESS DIALOG BOX
ACTION_ADDRECIPIENT	3	Add the recipients pointed by the Resolve Name property.
ACTION_DEL_RECIPIENT	4	Delete the recipient descriptor pointed by the recipient control.
ACTION_RECIP_SET	5	Set the Recipient of the current Working Message being used in the bound message control
ACTION_INSERTCUSTOM	6	Add a Custom Recipient to your Mail eXtension Recipient Control.
ACTION_ORIG_SET	7	Set the Message Originator (ONLY valid for VIM Gateway Session)

Remarks:
This Property is runtime write only. An Error event will be generated if the action fails.

Data Type:
Integer



AddRecipientClass

Example

Description:
Sets and returns the Default Recipient class used when adding new recipient descriptors.

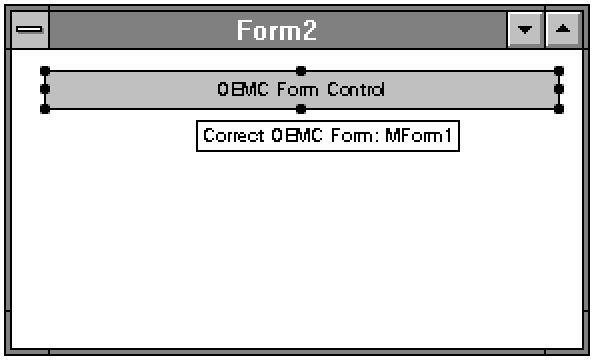
Usage:
e_RecipientClass=[form].MRecip. **AddRecipientClass**
[form].MRecip. **AddRecipientClass**=*e_RecipientClass*

Settings:

Value ID	Num.	Description
TO	1	Recipient Class = To
CC	2	Recipient Class = Cc
BCC	3	Recipient Class = Bcc

Remarks:
This value is used when adding new recipient to the control (Action, Methods).

Data Type:
ENUM



AddressCaption

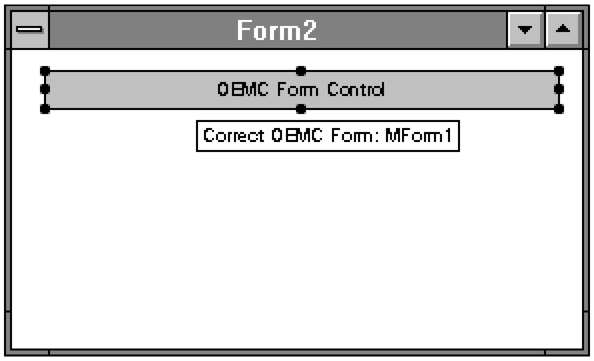
Example

Description:
Sets and returns the string being used when the Default Address DialogBox is used.

Usage:
hsz_CurrentCaption$=[form].MRecip. **AddressCaption**
[form].MRecip. **AddressCaption**=*hsz_NewCaption$*

Remarks:
If this property is an empty string, the **Address Book** caption is used by default. See Action property.

Data Type:
String



AddressEditNum

Example

Description:
Sets and returns the number of Edit Boxes that are used when the Address Dialog Box is used.

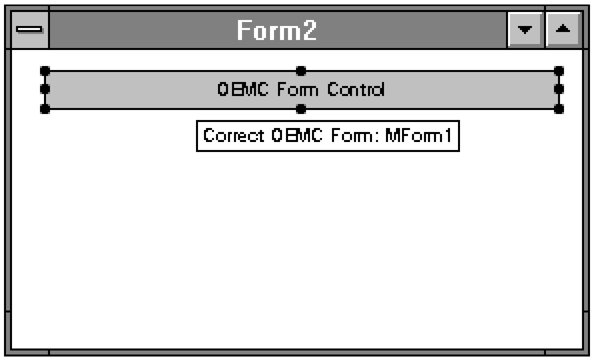
Usage:
i_EditNum=[form].MRecip. **AddressEditNum**
[form].MRecip. **AddressEditNum**=*i_EditNum*

Values:

Value	Num	Description
LIST_ONLY	0	Display only the Address List Box
EDIT_TO	1	Display 1 edit Box (To:)
EDIT_TO_CC	2	Display 2 edit boxes (To: and Cc:)
EDIT_TO_CC_BCC	3	Display 3 edit boxes (To:, Cc: and Bcc:)

Remarks:
See Action property.

Data Type:
Integer



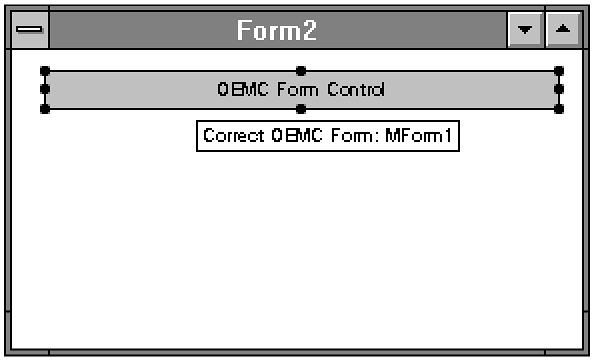
BindString

Description:
Returns the binding connection string

Usage:
hsz_BindingString$=[form].MRecip. **BindString**

Remarks:
The BindString string contains the Form and the name of the bound control

Data Type:
String



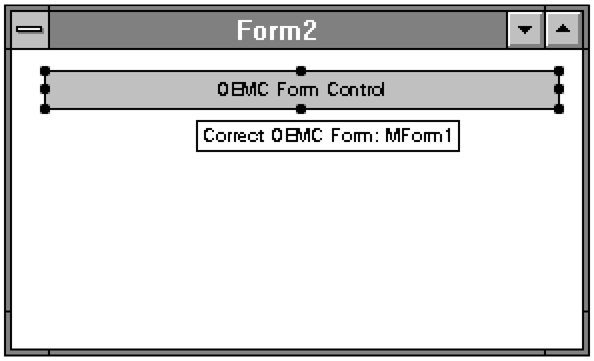
DetailModifiable

Description:
Set this property if the Recipient Control have to modify the recipient information when the ResolveName property is used.

Usage:
b_IsModify=[form].MRecip. **DetailModifiable**
[form].MRecip. **DetailModifiable**=*b_Modify*

Remarks:
If this property is FALSE, The ResolveName property will return an error if the recipients are not found in the address book.

Data Type:
BOOL



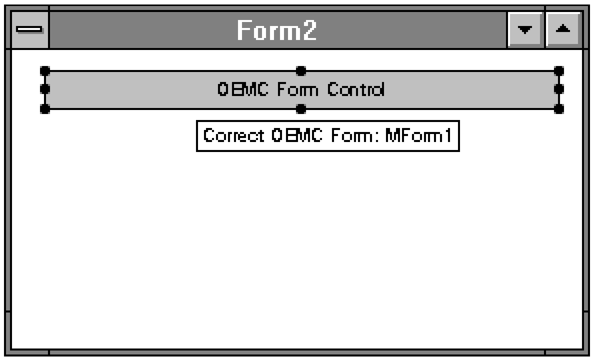
DisplayErrors

Description:
This property sets and returns the error handling state.
When setting the DisplayErrors property, the Recipient Control will Display a message box when an error occurs.

Usage:
*b_DisplayErrors=[form].MRecip.***DisplayErrors**
[form].MRecip. **DisplayErrors**=*b_DisplayErrors*

Remarks:
Regardless of this value, the ErrorNum and ErrorText properties are changed before an Error Event is generated

Data Type:
BOOL



ErrorNum

Example

Description:
This property returns the number of the last error.

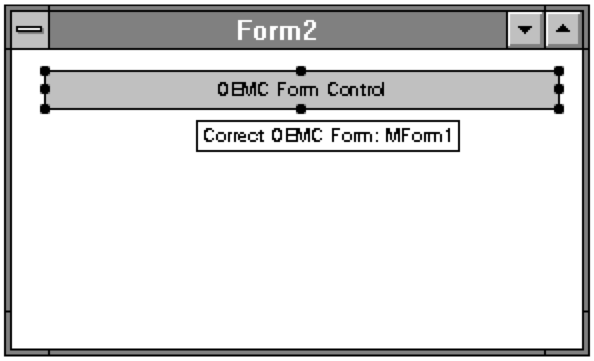
Usage:
i_ErrorNum%=[form].MRecip. **ErrorNum**

Values:

Error Code	Number	Description
ERROR_REC_NOTBOUND	1	The Recipient Control is NOT bound
ERROR_REC_MSGNOT_BOUND	2	The Bound Message Control is not correctly bound
ERROR_REC_EMPTY	3	The Recipient control is empty
ERROR_REC_SESS_NOCONNNECT	4	The indirect session control is not connected
ERROR_REC_BADCONTROL	5	Unable to clone MAILX recipient control
ERROR_REC_AMBIGUOUS	7	Ambiguous Recipients when Resolving address

Remarks:
This property is set to zero (0) before another action gets executed.

Data Type:
Integer



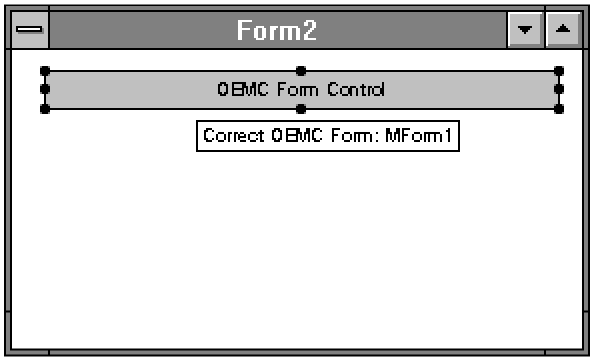
ErrorText

Description:
This properties contains the last error text.

Usage:
hsz_ErrorText$=[form].MRecip. **ErrorText**

Remarks:
This property is set regardless of the DisplayErrors property.

Data Type:
String



FetchRecipient

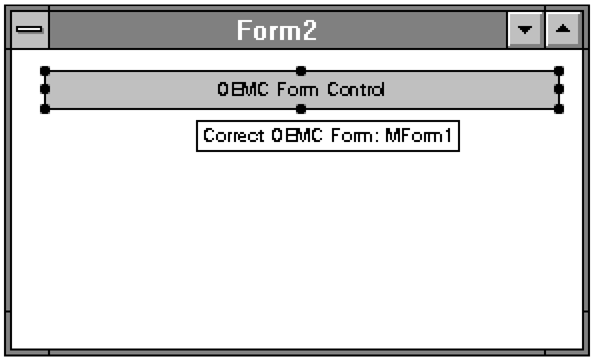
Example

Description:
Fetch or Flush the message recipients of the bound message control.

Usage:
b_IsFetched=[form].MRecip. **FetchRecipient**
[form].MRecip. **FetchRecipient**=*b_Fetch*

Remarks:
Return TRUE if the Recipient controls contains recipient descriptors, otherwise a FALSE is returned.

Data Type:
BOOL



FetchType

Example

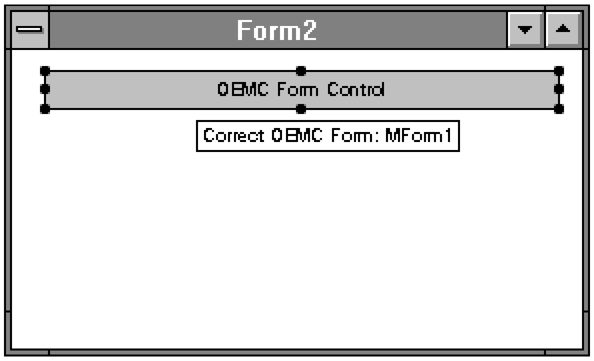
Description:
Set and returns the current type of recipients being fetched.

Usage:
i_FetchType%=[form].MRecip. **FetchType**
[form].MRecip. **FetchType**=*i_FetchType%*

Settings:

Value	Number	Description
Originator	0	Fetch the Originator recipient
Msg Recipients	1	Fetch the message recipients

Data Type:
Enum



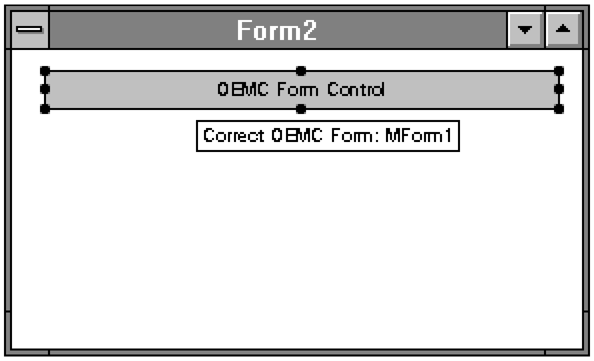
IsBinded

Description:
Returns the binding state of the Mail X Recipient Control.

Usage:
b_IsBinded=[form].MRecip. **IsBinded**

Remarks:
Returns TRUE if the control is successfully bound, otherwise a FALSE is returned

Data Type:
BOOL



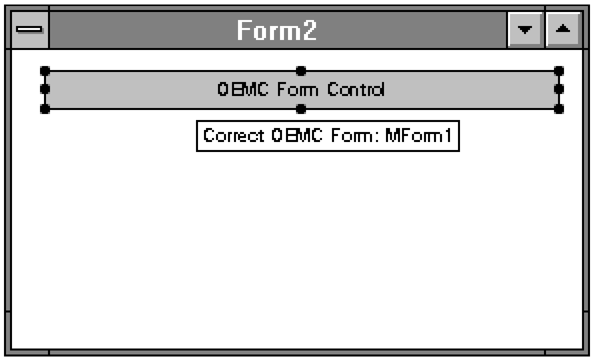
RecipientAddress

Description:
Sets and Return the current recipient address.

Usage:
hsz_RecipientAddress$=[form].MRecip. **RecipientAddress**
[form].MRecip. **RecipientAddress**=*hsz_RecipientAddress$*

Remarks:
returns an empty string if the Recipient control is empty or not fetched.

Data Type:
String



RecipientClass

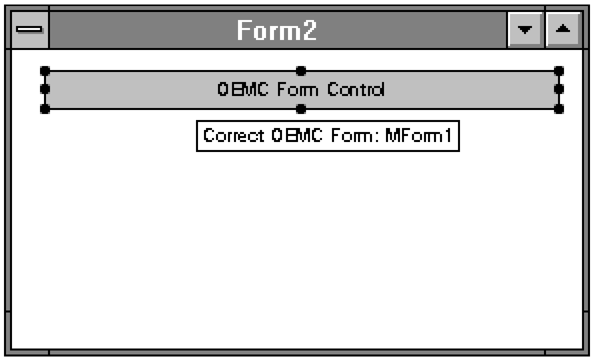
Description:
Sets and returns the current recipient class.

Usage:
i_RecipientClass&=[form].MRecip. **RecipientClass**
[form].MRecip. **RecipientClass**=*i_RecipientClass&*

Values:

Value	Number	Description
MAPI_ORIG	0	Originator
MAPI_TO	1	Recipient Class = To:
MAPI_CC	2	Recipient Class = Cc:
MAPI_BCC	3	Recipient Class = Bcc:

Data Type:
Integer



RecipientCount

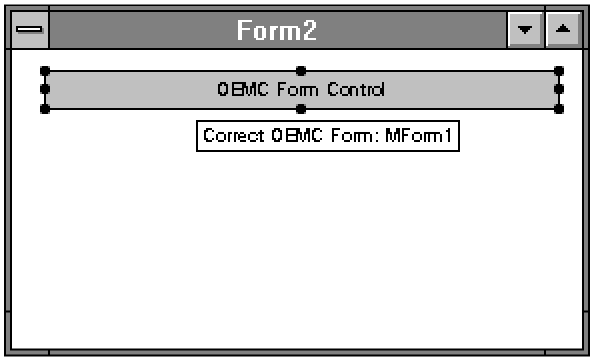
Example

Description:
Returns the number of Recipients available in the Recipient Control.

Usage:
i_RecipientCount%=[form].MRecip. **RecipientCount**

Remarks:
You need to Fetch or Add recipients to include new recipient descriptors in the Mail X Recipient Control

Data Type:
Long Integer



RecipientName

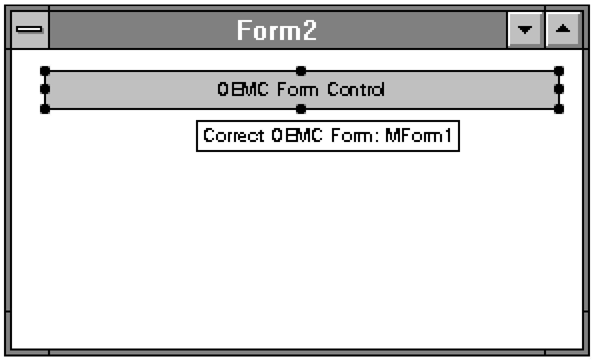
Example

Description:
Sets and Return the current recipient name.

Usage:
hsz_RecipientName$=[form].MRecip. **RecipientName**
[form].MRecip. **RecipientName**=*hsz_RecipientName$*

Remarks:
returns an empty string if the Recipient control is empty or not fetched.

Data Type:
String



RecipientNum

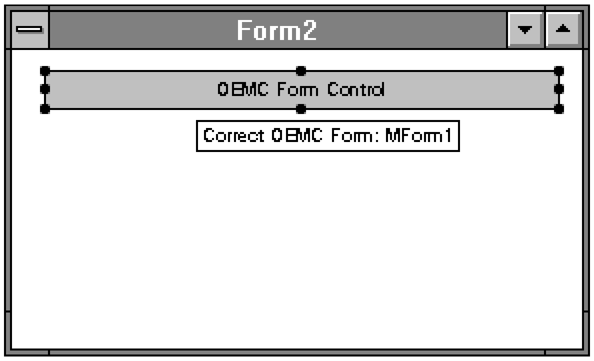
Example

Description:
This property sets and returns the current recipient descriptor number being used by the control

Usage:
l_CurrentRecipientNum&=[form].MRecip. **RecipientNum**
[form].MRecip. **RecipientNum**=*l_CurrentRecipientNum&*

Remarks:
If you set an invalid recipient number, the recipient control points to the last valid descriptor.

Data Type:
Long Integer



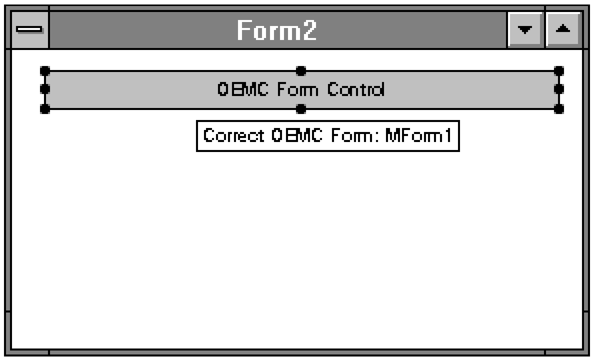
ResolveDialog

Description:
Sets and returns the current Resolve Dialog state of the control

Usage:
b_DisplayResolveDialog=[form].MRecip. **ResolveDialog**
[form].MRecip. **ResolveDialog**=*b_DisplayResolveDialog*

Remarks:
If TRUE, a Resolve Dialog Box will be shown when resolving recipient names, Otherwise the address name being resolved will be deleted from the list.

Data Type:
BOOL



ResolveName

Example

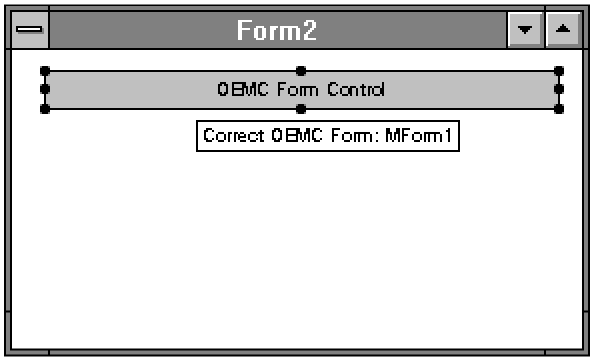
Description:
This property resolves a mail recipient's name (as entered by a user) to an unambiguous address list entry. This property return the corrected address list.

Usage:
[form].MRecip. **ResolveName**=*hsz_AddressList$*
hsz_CorrectedAddressList=[form].MRecip. **ResolveName**

Remarks:
When setting this property, you can pass a list of address name to be resolved:
Name1;Name2;Name3....

Optionally , you can prompt the user to choose between ambiguous entries if necessary (Set the Resolve Dialog property).

Data Type:
String

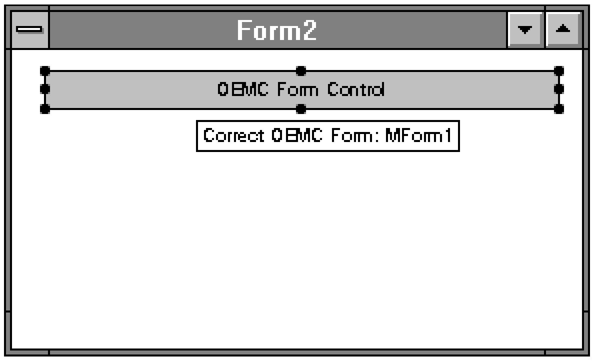


Error Event

Description:
Occurs whenever an errors is generated by the Recipient Control.

Syntax:
Sub MRecip_Error ()

Remarks:
If you Set the DisplayErrors property, a Message Box will be displayed before the Error Event occurs.



AddItem Method

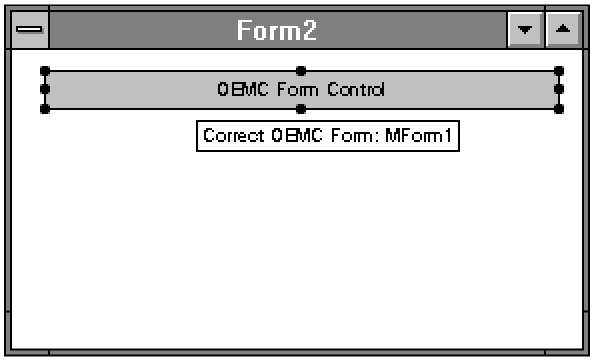
Description:
Adds a Recipient to the Mail X Recipient control.

Usage:
[form]. MRecip. **AddItem** AddressNameList

Remarks:
This Method resolves a mail recipient's name (as entered by a user) to an unambiguous address list entry, optionally prompting the user to choose between ambiguous entries if necessary (See ResolveDialog Property). You can ADD a list of address name to be included:

Name1;Name2;Name3....

The index values has no meaning.

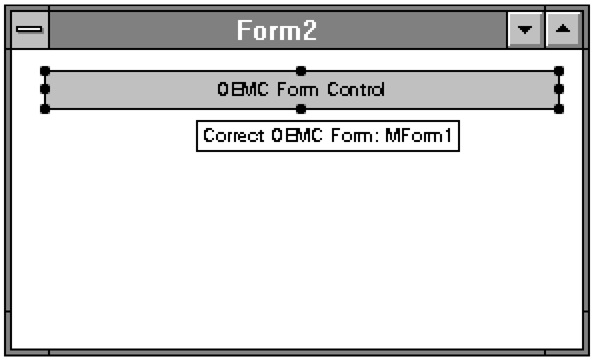


Remove Method

Description:
Remove the specified recipient descriptor.

Usage:
[form]. MRecip. **RemoveItem** index%

Remarks:
The index value indicates the file descriptor number (or position) that will be deleted.

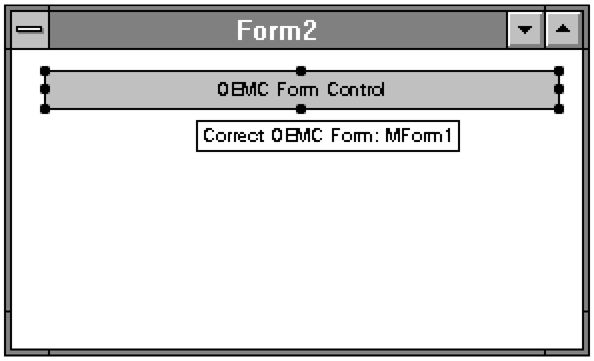


Clear Method

Example

Description:
Flush the current Recipient descriptor list.

Usage:
[form]. MRecip. **Clear**

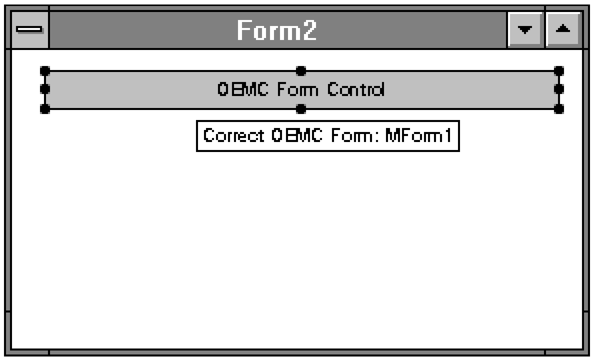


Refresh Method

Example

Description:
Set the current recipient descriptor list to the bound Mail X Message Control.

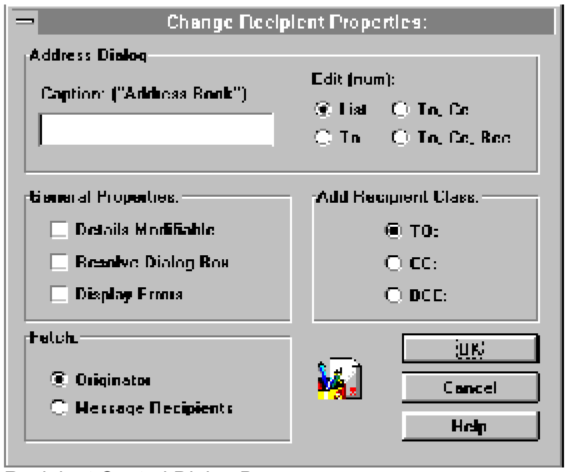
Usage:
[form]. MRecip. **Refresh**



Changing Recipient Control Properties

You can change the Mail X Recipient properties with a click on the Help Label button.

The Mail X Recipient Control Dialog Box is only available at Design time.


Change Recipient Properties:

Address Dialog

Caption: ("Address Book")

Edit (num): [✓ List] To To, Cc To, Cc, Bcc

General Properties:

- [ ] Details Modifiable
- [ ] Resolve Dialog Box
- [ ] Display Errors

Add Recipient Class: [✓ TO:] CC: BCC:

Fetch: [✓ Originator] Message Recipients

Recipient Control Dialog Box

Properties:

- About
- Action
- AddRecipientClass
- AddressCaption
- AddressEditNum
- BindString
- DetailModifiable
- DisplayErrors
- ErrorNum
- ErrorText
- FetchRecipient
- FetchType
- Height
- Index
- IsBinded
- Left
- Name
- ObjRef
- RecipientAddress
- RecipientClass
- RecipientCount
- RecipientName
- RecipientNum
- ResolveDialog
- ResolveName
- Tag
- Top
- Width

Methods:

- AddItem
- BindWith
- RemoveItem
- Clear
- Refresh

Events:

- Error

Example

The following code uses a MailX Session control, Message Control, Recipient Control and a button on a form. The code display the Address Book.

Visual Basic Code:

```
Sub BtnAddressDialog_Click ()
    MReci1.Action = ACTION_ADDRESS
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control,   Edit Box and a button on a form. The code resolves the Address Name and, if it is found in the Book directory, It will be added to the Recipient Control

Visual Basic Code:

```
Sub BtnResolve_Click ()
    MReci1.ResolveName = Text1.Text
    If MReci1.ResolveName <> "" Then
        MReci1.Action = ACTION_ADDRECIPIENT
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control,   Edit Box and a button on a form. The code resolves the Address Name and, if it is found in the Book directory, It will be added to the Recipient Control as CC recipient

Visual Basic Code:

```
Sub BtnResolve_Click ()
    MReci1.ResolveName = Text1.Text
    If MReci1.ResolveName <> "" Then
        MReci1.AddRecipientClass = CLASS_CC
        MReci1.Action = ACTION_ADDRECIPIENT
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control and a button on a form. The code display the Address Book with a custom Title.

Visual Basic Code:

```
Sub BtnAddressDialog_Click ()
    MReci1.AddressCaption = My Own Caption ""
    MReci1.Action = ACTION_ADDRESS
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control and a button on a form. The code display the Address Book with the recipient List ONLY

Visual Basic Code:

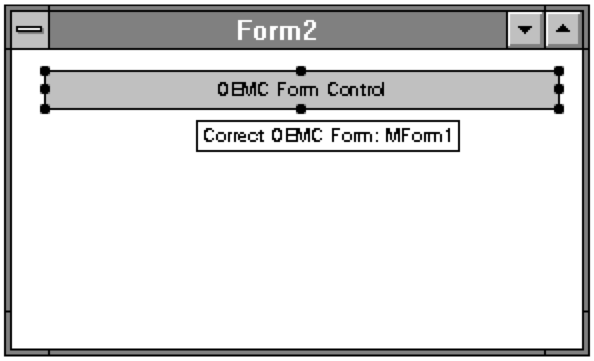
```
Sub BtnAddressDialog_Click ()
    MReci1.AddressEditNum =0
    MReci1.AddressCaption = My Own Caption
    MReci1.Action = ACTION_ADDRESS
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control, Edit Box and a button on a form. The code read the first message and fetch the Originator Recipient, The Recipient Name is shown in the Edit Box

Visual Basic Code:

```
Sub Command1_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    If MMsg1.ErrorNum = 0 Then
        MReci1.FetchType = ORIGINATOR
        MReci1.FetchRecipient = True
        MReci1.RecipientNum = 1
        Text1.Text = MReci1.RecipientName
    End If
End Sub
```



Mail eXtension Custom Address

Lotus cc:Mail Example Lotus Notes Example

Overview:

Mail eXtension Custom Address are useful when sending messages to any gateway (*or remote PostOffice*) defined in your Mail Server. You only have to specify your Recipient Address and Mail eXtension will try to send the e-mail through your PostOffice.

Mail eXtension will generate an error if your Recipient Address is not supported by your PostOffice

How to use Custom Recipient Address?

To create a Custom Address, you have to add a blank recipient descriptor into your Recipient Control:

```
MReci1.Action = ACTION_INSERTCUSTOM
```

When you insert the Custom Recipient, You will have to set the RecipientAddress in order to specify your Custom Address:

(*Lotus cc:Mail example)*
```
MReci1.Action = ACTION_INSERTCUSTOM
MReci1.RecipientAddress = Mark Green at PostOffice
MReci1.RecipientName = Mark Green at PostOffice
```

(*Lotus Notes example)*
```
MReci1.Action = ACTION_INSERTCUSTOM
MReci1.RecipientAddress = @Mgreen@tpd.com
MReci1.RecipientName = @Mgreen@tpd.com
```

(*Microsoft Exchange Server)*
```
MReci1.Action = ACTION_INSERTCUSTOM
MReci1.RecipientAddress = Mgreen@tpd.com
MReci1.RecipientName = Mgreen@tpd.com
```

NOTE: You have to indicate the Complete address recipient in order to send your Mail Message

Example

The following code uses a MailX Session control, Message Control, Recipient Control, 2 EditBoxes and a button on a form. The code will send a Simple e-Mail message to the external Lotus cc:Mail PostOffice INET-PO

Visual Basic Code:

```
Sub Command1_Click ()
    If MSess1.Logon = True Then
        MReci1.Clear
        MReci1.AddRecipientClass= CLASS_TO
        MReci1.Action = ACTION_INSERTCUSTOM
        MReci1.RecipientAddress = "Jimmy@company.com at INET-PO"
        MReci1.RecipientName = "Jimmy@company.com at INET-PO"
        MMsg1.Action = ACTION_NEW
        MMsg1.WorkingMsg = COMPOSE_MSG
        MReci1.Refresh
        MMsg1.Subject = Text1.Text
        MMsg1.NoteText = Text2.Text
        MMsg1.Action = ACTION_SENDMSG
        If MMsg1.ErrorNum = 0 Then
                MsgBox "Message has been sent"
        Else
                MsgBox "Error Sending The Mail message"
        End If
     Else
        MsgBox "No Active Session available"
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control and a button on a form. The code will fetch the first message and returns the Number of recipients

Visual Basic Code:

```
Sub Command3_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    If MMsg1.ErrorNum = 0 Then
        MReci1.FetchType = MSG_RECIPIENTS
        MReci1 = MMsg1
        MsgBox "Number of Recipients=" + Str$(MReci1.RecipientCount)
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control, 2 EditBoxes and a button on a form. The code will send a Simple e-Mail message through the Lotus Notes Server

Visual Basic Code:

```
Sub Command1_Click ()
    If MSess1.Logon = True Then
        MReci1.Clear
        MReci1.AddRecipientClass= CLASS_TO
        MReci1.Action = ACTION_INSERTCUSTOM
        MReci1.RecipientAddress = "@Jimmy@company.com"
        MReci1.RecipientName = "@Jimmy@company.com "
        MMsg1.Action = ACTION_NEW
        MMsg1.WorkingMsg = COMPOSE_MSG
        MReci1.Refresh
        MMsg1.Subject = Text1.Text
        MMsg1.NoteText = Text2.Text
        MMsg1.Action = ACTION_SENDMSG
        If MMsg1.ErrorNum = 0 Then
                MsgBox "Message has been sent"
        Else
                MsgBox "Error Sending The Mail message"
        End If
     Else
        MsgBox "No Active Session available"
    End If
End Sub
```

Mail X File Control

Properties Methods Events

Overview:

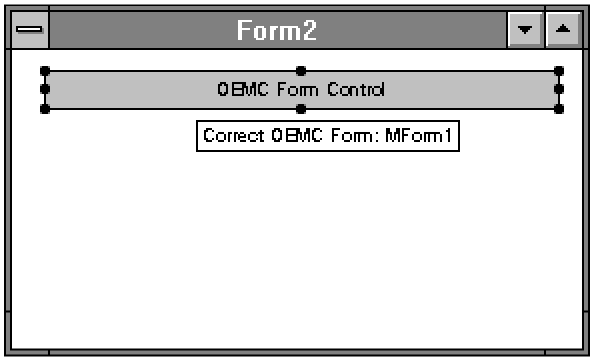
The Mail X File Control is used to handle the attachments associated with a message contained in a Mail X Message Control. You can query or add new file attachments to the current message contained in the Message Control.

The File Control only handle information relative to the file attachments of a message (*name, path, position, flags, etc.*). To obtain or set the file attachments of a mail message, BIND the Mail X File Control to the Message Control.

How to retrieve File Attachments?

Click your Right button and change the File Control properties with a Mail X File Control DialogBox

Note: You must have a Mail X Form Control before adding
Mail X File Control on your Visual Basic Form



Action

Example

Description:
This property set a new Action in your Mail X File Custom Control.

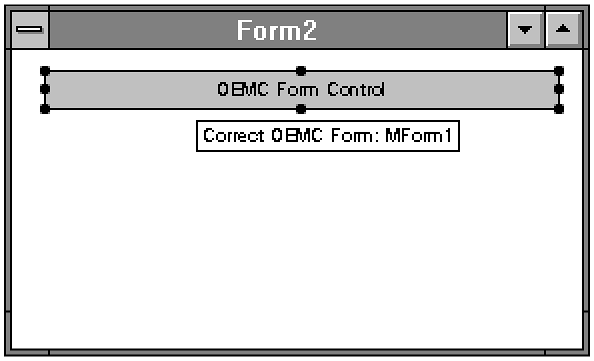
Usage:
[form].MFile. **Action**=i_NewAction

Values:

Action ID	Number	Description
ACTION_DELETEFILE	1	Remove the current file entry. File Position and Count property are modified
ACTION_REMOVEFILE	2	Remove the current file entry and delete the temporary file created when the message was read
ACTION_ADDFILE	3	Add a New File attachment.. The Display Dialog Property must be set
ACTION_FILESET	4	Set the File attachment of the current Working Message being used in the bound message control
ACTION_DELETE_TEMPFILE	5	Delete the temporary file created when the message was read

Remarks:
This Property is runtime write only. An Error event will be generated if the action fails.

Data Type:
Integer



AttachmentPosition

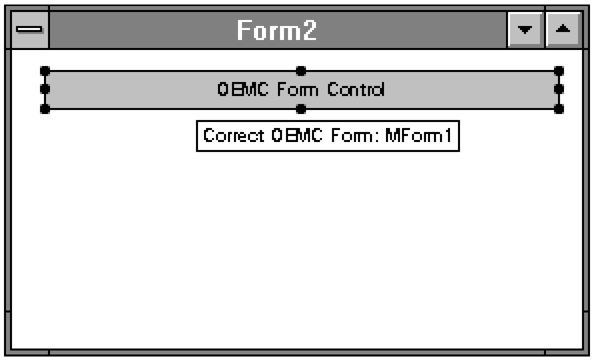
Example

Description:
Sets and returns the position of the current attachment in the message body.

Usage:
i_AttachPosition%=[form].MFile. **AttachmentPosition**
[form].MFile. **AttachmentPosition**=*i_AttachPosition%*

Remarks:
Attachments replace the character found at a certain position in the message body. Applications may not place two attachments in the same location within a message, and attachments may not be placed beyond the end of the message body.

Data Type:
Long Integer



Correct OEMC Form: MForm1

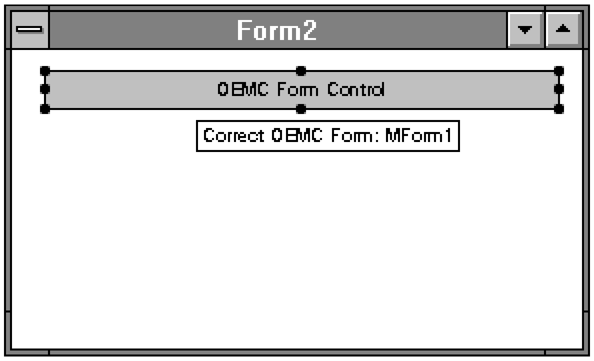
BindString

Description:
Returns the binding connection string

Usage:
hsz_BindingString$=[form].MFile. **BindString**

Remarks:
The BindString string contains the Form and the name of the bound control

Data Type:
String



DisplayDialog

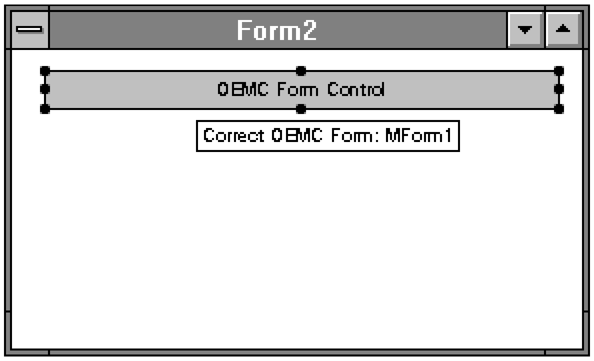
Example

Description:
Sets and returns the Display Dialog state

Usage:
b_DisplayDialog=[form].MFile. **DisplayDialog**
[form].MFile. **DisplayDialog**=*b_DisplayDialog*

Remarks:
When this property is set, a add file dialog box (*Open File Common Dialog*) will be shown when file attachments are added to the control.

Data Type:
BOOL



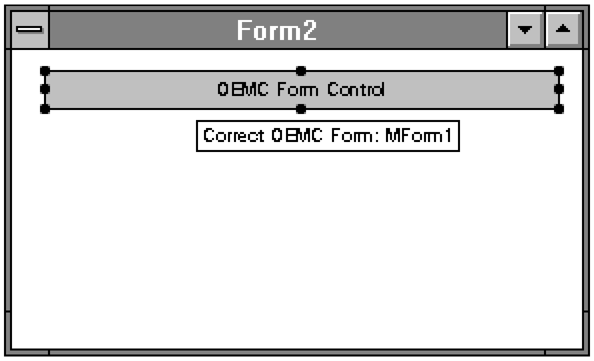
DisplayErrors

Description:
This property sets and returns the error handling state.
When setting the DisplayErrors property, the File Control will Display a message box when an error occurs.

Usage:
*b_DisplayErrors=[form].MFile.***DisplayErrors**
[form].MFile. **DisplayErrors**=*b_DisplayErrors*

Remarks:
Regardless of this value, the ErrorNum and ErrorText properties are changed before an Error Event is generated

Data Type:
BOOL



ErrorNum

Description:
This property returns the number of the last error.

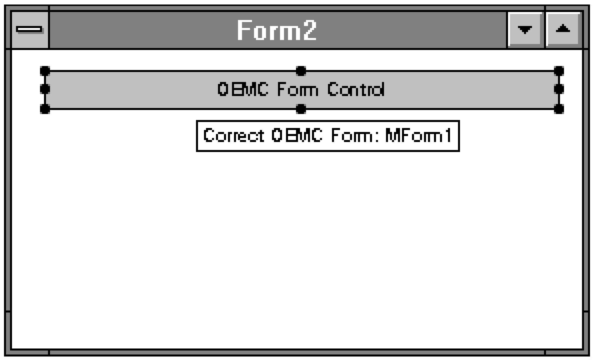
Usage:
i_ErrorNum%=[form].MFile. **ErrorNum**

Values:

Error Code	Number	Description
ERROR_DELETE_TEMP	1	Unable to delete the Temporary File attachment
ERROR_FILE_EMPTY	2	Unable to execute the operation because the File Control is empty
ERROR_FILE_NOT_BOUND	3	The File Control is NOT bound
ERROR_MSGNOT_BOUND	4	The bound Message Control is not correctly bound
ERROR_FILE_BADCONTROL	5	Error while cloning MailX File Control
ERROR_FILE_UNABLEADD	7	Unable to add file attachment

Remarks:
This property is set to zero (0) before another action gets executed.

Data Type:
Integer



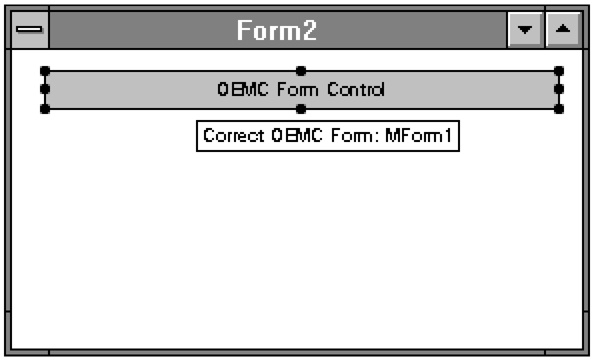
ErrorText

Description:
This properties contains the last error text.

Usage:
hsz_ErrorText$=[form].MFile. **ErrorText**

Remarks:
This property is set regardless of the DisplayErrors property.

Data Type:
String



FetchFile

Example

Description:
This property is set to fetch or flush the file attachment of the bound message control

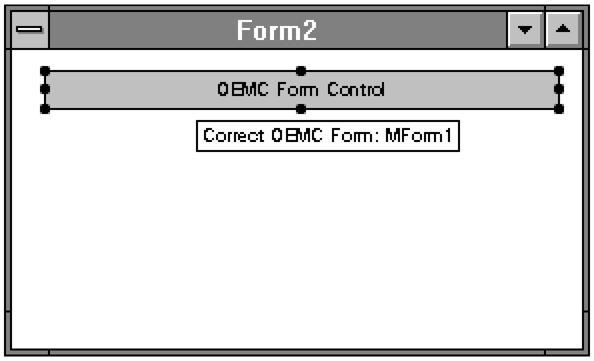
Usage:
b_FetchFile=[form].MFile. **FetchFile**
[form].MFile. **FetchFile**=*b_FetchFile*

Settings:

Values	Num	Description
TRUE	-1	Fetch the file attachment of the bound message
FALSE	0	Flush the current file attachment available in the control.

Remarks:
This property returns TRUE, whenever a file attachment descriptor is available in the control.

Data Type:
BOOL



FileCount

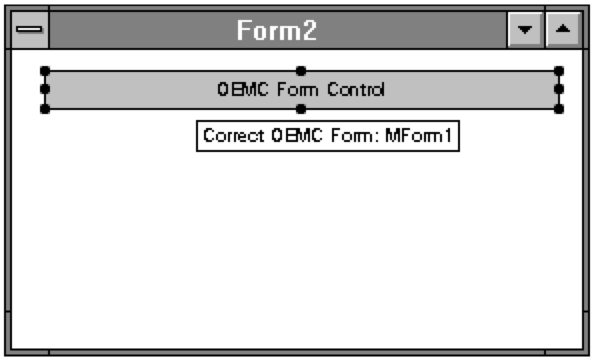
Example

Description:
Returns the number of File attachment available in the File Control.

Usage:
i_FileCount%=[form].MFile. **FileCount**

Remarks:
You need to Fetch or Add file attachment to include new file descriptors in the File Control

Data Type:
Long Integer



FileFlags

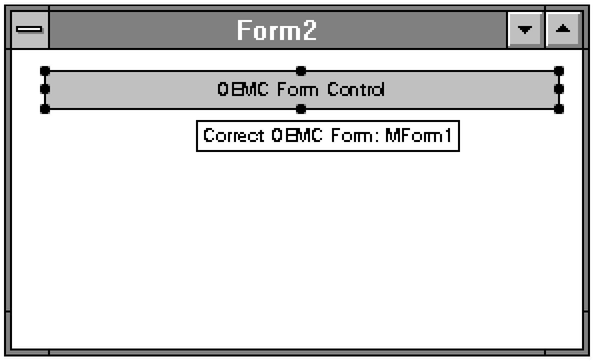
Description:
Sets and returns the Flags of the current File Attachment.

Usage:
l_CurrentFlags&=[form].MFile. **FileFlags**
[form].MFile. **FileFlags**=*l_NewFlags&*

Settings:

Value	Number	Description
NO_FLAGS	0	No Flags defined
MAPI_OLE	1	OLE Object
MAPI_OLE_STATIC	2	Static OLE Object

Data Type:
Long Integer



FileName

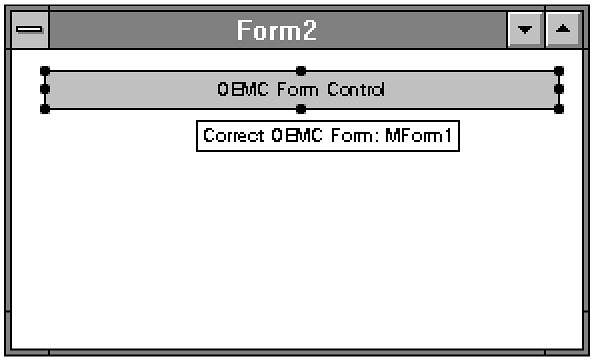
Example

Description:
Returns the name of the current file attachment.

Usage:
hsz_FileName$=[form].MFile. **FileName**

Remarks:
The result may be an empty string if no attachment are available in the controls.

Data Type:
String



FileNum

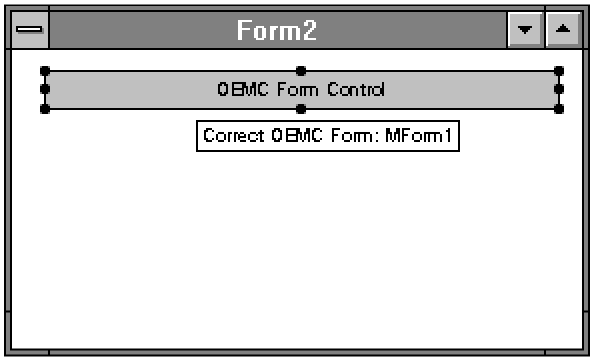
Example

Description:
This property sets and returns the current file descriptor number being used by the control

Usage:
l_CurrentFileNum&=[form].MFile. **FileNum**
[form].MFile. **FileNum**=*l_CurrentFileNum&*

Remarks:
If you set an invalid file number, the File control points to the last valid descriptor.

Data Type:
Long



FilePath

Description:

This property returns the path and file name of the temporary file attachment.

Usage:

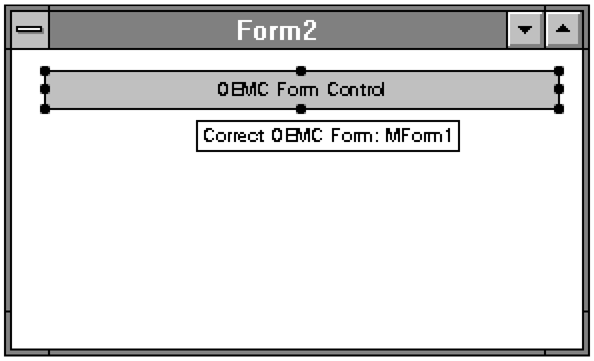
hsz_FilePath$=[form].MFile. **FilePath**

Remarks:

The result may be an empty string if no attachment are available in the controls.

Data Type:

String



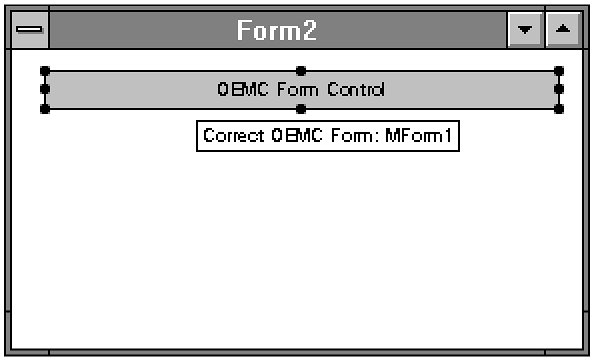
IsBinded

Description:
Returns the binding state of the Mail X File Control.

Usage:
b_IsBinded=[form].MFile. **IsBinded**

Remarks:
Returns TRUE if the control is successfully bound, otherwise a FALSE is returned

Data Type:
BOOL

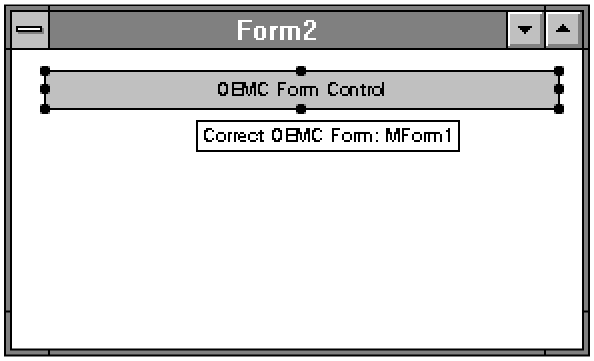


Error Event

Description:
Occurs whenever an errors is generated by the File Control.

Syntax:
Sub MFile_Error ()

Remarks:
If you Set the DisplayErrors property, a Message Box will be displayed before the Error Event occurs.



AddItem Method

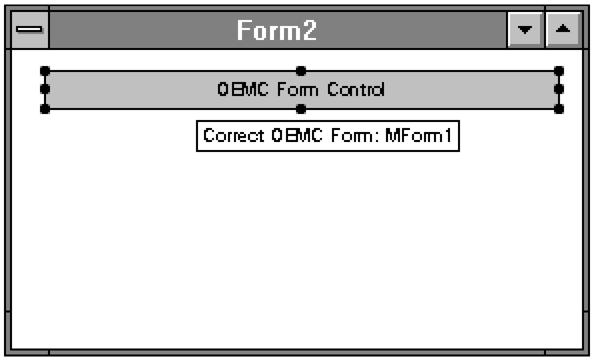
Example

Description:
Adds a File attachment to the Mail X File control.

Usage:
[form].MFile. **AddItem** FileName

Remarks:
If the FileName is not found; The Add File Dialog Box will be displayed.

Remarks:
The index values has no meaning.

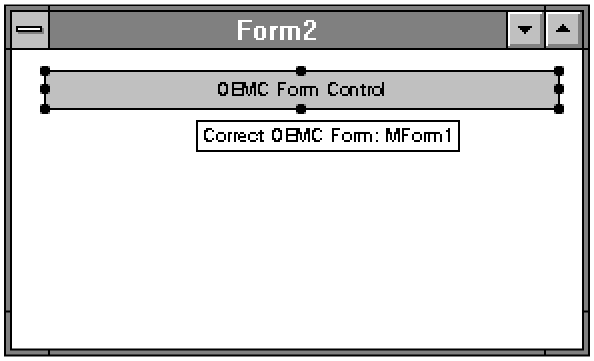


Remove Method

Description:
Remove the specified file descriptor and its temporary file.

Usage:
[form].MFile. **RemoveItem** index%

Remarks:
The index value indicates the file number that will be deleted.

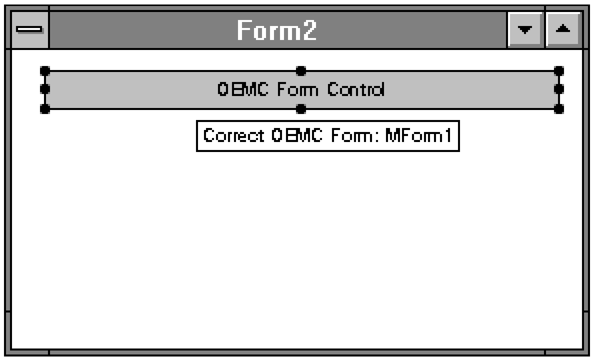


Clear Method

Example

Description:
Flush the current File descriptor list.

Usage:
[form].MFile. **Clear**

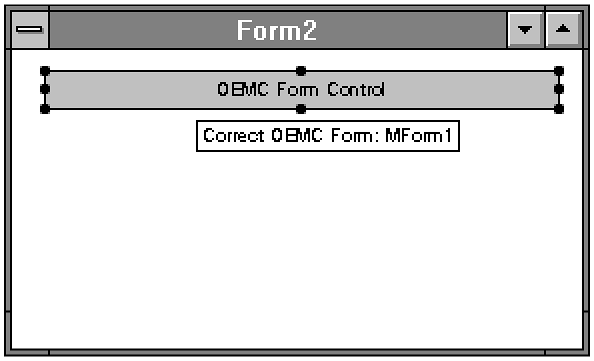


Refresh Method

Example

Description:
Set the current File descriptor list to the bound Mail X Message Control.

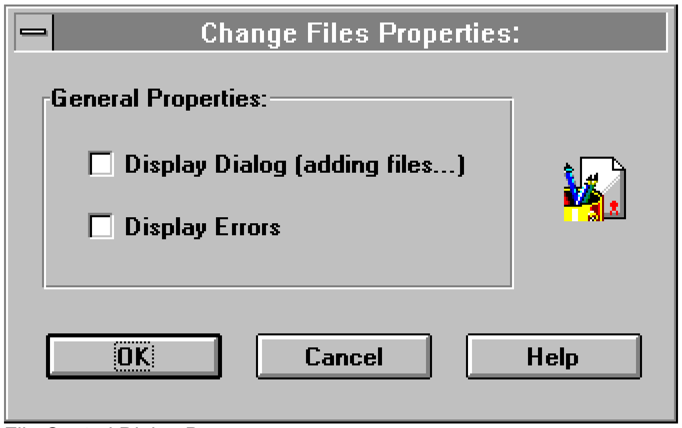
Usage:
[form].MFile. **Refresh**



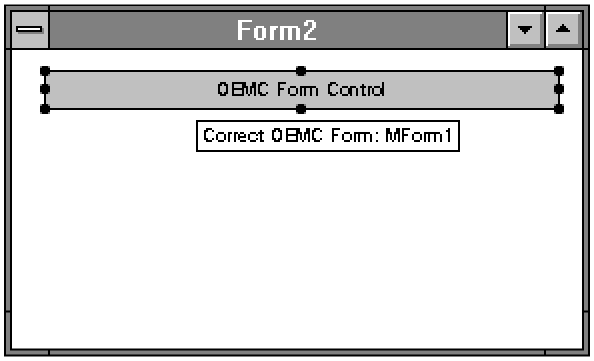
Changing File Control Properties

You can change the Mail X file properties with a click on the Help Label button.

The Mail X File Control Dialog Box is only available at Design time.



File Control Dialog Box



Retrieving File Attachments

When reading Mail Messages, The 'Reading Properties' will change the behaviour of your Mail X Message control.

Reading Properties:
-BodyAsFile
-EnvelopeOnly
-MarkAsRead
-SuppressAttach

By default, Mail X Message Control sets:
BodyAsFile=FALSE
EnvelopeOnly=FALSE
SupressAttach=TRUE
MarkAsRead=TRUE

Set 'EnvelopeOnly' when you don't want MailX to copy file attachments to temporary files or return the note text. All other message information (except for temporary filenames) is returned. Setting this flag usually reduces the processing time required for the function.

Set 'SupressAttach' when you don't want MAPIReadMail (SMAPI function) to copy file attachments, but instead just return message text. This flag is ignored if 'EnvelopeOnly' is set. The flag should reduce the time required by the MAPIReadMail function.

Set 'BodyAsFile' when you want the message body written to a temporary file and added to the attachment list as the first attachment, instead of returning a pointer to the message body.

If setting SupressAttach=TRUE (default behaviour), you will be able to 'read' the File Attachment Name but the temporary File will not be created (The MFile.FilePath will return an Empty string); In order to 'create' the temporary File Attachments, you will have to set SupressAttach=FALSE & EnvelopeOnly=FALSE when reading your Mail Message.

NOTE: When Creating TEMPORARY File Attachment, You will have to remove them manually. The Mail eXtension MFILE control contains some actions to delete the temporary File: (MFile1.Action=ACTION_REMOVEFILE).

Properties:

- About
- Action
- AttachmentPosition
- BindString
- DisplayDialog
- DisplayErrors
- ErrorNum
- ErrorText
- FetchFile
- FileCount
- FileFlags
- FileName
- FileNum
- FilePath
- Height
- Index
- IsBinded
- Left
- Name
- ObjRef
- Tag
- Top
- Width

Events:

- Error

Methods:

- AddItem
- BindWith
- RemoveItem
- Clear
- Refresh

Example

The following code uses a MailX Session control, Message Control, Recipient Control, File Control and a button on a form. The code will compose a new message and set a file Attachment

Visual Basic Code:

```
Sub Command2_Click ()
    If MSess1.Logon = True Then
        MReci1.Clear
        MReci1.ResolveName = szAddress.Text
        If MReci1.ResolveName = "" Then
            MsgBox "Recipient NOT FOUND"
        Else
            MReci1.Action = ACTION_ADDRECIPIENT
            MMsg1.Action = ACTION_NEW
            MMsg1.WorkingMsg = COMPOSE_MSG
            MReci1.Action = ACTION_RECIP_SET
            MMsg1.Subject = szSubject
            MMsg1.NoteText = szNoteText
            MFile1.Clear
            MFile1.AddItem szFileName.Text
            If MFile1.FileCount = 0 Then MsgBox "Error Setting File Attachment"
            MFile1.Action = ACTION_FILESET

            MMsg1.Action = ACTION_SENDMSG

            If MMsg1.ErrorNum = 0 Then
                MsgBox "Message has been sent"
            Else
                MsgBox "Error Sending The Mail message"
            End If
        End If
    Else
        MsgBox "No Active Session available"
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, Recipient Control, File Control and a button on a form. The code will compose a new message and set a file Attachment

Visual Basic Code:

```
Sub Command2_Click ()
    If MSess1.Logon = True Then
        MReci1.Clear
        MReci1.ResolveName = szAddress.Text
        If MReci1.ResolveName = "" Then
            MsgBox "Recipient NOT FOUND"
        Else
            MReci1.Action = ACTION_ADDRECIPIENT
            MMsg1.Action = ACTION_NEW
            MMsg1.WorkingMsg = COMPOSE_MSG
            MReci1.Action = ACTION_RECIP_SET
            MMsg1.Subject = szSubject
            MMsg1.NoteText = szNoteText
            MFile1.Clear
            If MsgBox("Display AddFile Dialog?", 4, "Mail eXtension") = 6 Then
                MFile1.DisplayDialog = True
            Else
                MFile1.DisplayDialog = False
            End If
            MFile1.AddItem szFileName.Text
            If MFile1.FileCount = 0 Then MsgBox "Error Setting File Attachment"
            MFile1.Refresh

            MMsg1.Action = ACTION_SENDMSG

            If MMsg1.ErrorNum = 0 Then
                MsgBox "Message has been sent"
            Else
                MsgBox "Error Sending The Mail message"
            End If
        End If
    Else
        MsgBox "No Active Session available"
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, File Control and a button on a form.
The code will fetch the first message and returns the Number of File Attachments

Visual Basic Code:

```
Sub Command3_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    If MMsg1.ErrorNum = 0 Then
        MFile1 = MMsg1
        MsgBox "Number of Files=" + Str$(MFile1.FileCount)
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, File Control and a button on a form. The code will fetch the first message and returns the Number of File Attachments

Visual Basic Code:

```
Sub Command3_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    If MMsg1.ErrorNum = 0 Then
        MFile1.FetchFile = True
        MsgBox "Number of Files=" + Str$(MFile1.FileCount)
    End If
End Sub
```

Example

The following code uses a MailX Session control, Message Control, File Control and a button on a form. The code will add two File Attachments and will set the Attachment position

Visual Basic Code:

```
Sub Command2_Click ()
    MFile1.Clear
    MFile1.AddItem "C:\CONFIG.SYS"
    MFile1.FileNum = 1
    MFile1.AttachmentPosition = 1
    MFile1.AddItem "C:\AUTOEXEC.BAT"
    MFile1.FileNum = 2
    MFile1.AttachmentPosition = 2
    MsgBox "File Count=" + Str$(MFile1.FileCount)
End Sub
```

Example

The following code uses a MailX Session control, Message Control, File Control and a button on a form. The code will fetch the first message and returns the FileName of the first FileAttachment available in your message

Visual Basic Code:

```
Sub Command3_Click ()
    MMsg1.Action = ACTION_FINDFIRST
    If MMsg1.ErrorNum = 0 Then
        MFile1 = MMsg1
        If MFile1.FileCount>0 then
            MFile1.FileNum = 1
            MsgBox MFile1.FileName
        Endif
    End If
End Sub
```