Help for Mail

Properties Events Methods Frequently Asked Questions

Registration Information

Order Form

Getting Custom Controls Written

Licensing Information

Description

Mail VBX/OCX is the only control in the VB market which is fully MIME compliant. It is really four controls in one. It provides SMTP and POP3. It also can send and receive MIME compliant messages with unlimited attachments. And, it supports encoding/decoding of Base64, Binhex40, UU, and Quoted-Printable.

Mabry Software's Mail meets the complete specifications for MIME (RFC1521 and 1522), POP3 (RFC1725), and SMTP (RFC0821).

Mail can also be used independently as a general purpose encoder and decoder control, or in combination with other controls which need encoding /decoding capability.

The Mail control comes with a sample program written in VB that is nearly a complete mail client. The sample program shows how to use all of the major features (and most of the minor ones) of Mail.

File Name

MMAIL1.VBX, MMAIL32.OCX

ActiveX / OCX Object Name

Mabry.MailCtrl

ActiveX Compatility

VB 4.0 (32-bit) and 5.0, Access 95, Access 97, Visual FoxPro 5.0, Delphi

ActiveX Built With

Microsoft Visual C++ v4

ActiveX - Required DLLs

MFC40.DLL (October 6th, 1995 or later)
OLEPRO32.DLL (October 6th, 1995 or later)
MSVCRT40.DLL (September 29th, 1995 or later)

VBX Object Type

mMail

VBX Compatibility

VB 2.0, 3.0 and 4.0 (16-bit)

VBX Built With

Microsoft Visual C++ v1.5

About Error Handling

There are two types of errors. The first type of error is thrown when a property is set or during the time between calling a method and the time the call returns. The second type of error is thrown only when Blocking is False and an error occurs during the asynchronous operation of the a method.

The first type of error must be handled using On Error trapping. The second type of error results in firing the AsyncError event.

So, complete error handling when Blocking is False requires On Error trapping during all method invocations and responding to firings of the AsyncError event. When Blocking is True then only On Error trapping is required.

Despite the extra work required to handle errors when Blocking is False it is recommended that this approach be taken with any application which has a user interface.

Important Note

The Blocking property must not be changed while a connection is established. You should only change the Blocking property when the control is not connected to a server. Changing the Blocking property while connected could cause some strange behavior in the control.

Distribution Note When you develop and distribute an application that uses this control, you should install the control file into the user's Windows SYSTEM directory. The control file has version information built into it. So, during installation, you should ensure that you are not overwriting a newer version.

Close

Mail Properties

Properties that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*__Action__ Property
*__BCC__ Property
*__Blocking__ Property
*__Body__ Property
*__BodyCount__ Property
*__Buffer__ Property
*__CC__ Property
*__ConnectType__ Property
*__ContentDescription__ Property
*__ContentDisposition__ Property
*__ContentID__ Property
*__ContentSubtype__ Property
*__ContentSubtypeParameters__ Property
*__ContentTransferEncoding__ Property
*__ContentType__ Property
*__Date__ Property
*__Debug__ Property
*__DstFilename__ Property
*__EMailAddress__ Property
*__Flags__ Property
*__From__ Property
*__Headers__ Property
*__HeadersCount__ Property
*__HeaderText__ Property
*__Host__ Property
*__LastError__ Property
*__Lines__ Property
*__LogonName__ Property
*__LogonPassword__ Property
*__MessageClass__ Property
*__MessageID__ Property
*__MultipartBoundary__ Property
*__Part__ Property
*__Parts__ Property
*__PopMessageCount__ Property
*__PopMessageSizes__ Property
*__PopPort__ Property
*__ReadData__ Property
*__SmtpPort__ Property

*__SrcFilename__ Property
*__Subject__ Property
*__Timeout__ Property
*__To__ Property
*__Version__ Property
*__WriteData__ Property

Close

Mail Events

Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*__AsyncError__ Event

*__Debug__ Event

*__Done__ Event

*__Progress__ Event

Close

Mail Methods

Methods that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

*__Abort__ Method
*__AboutBox__ Method
*__Ascend__ Method
*__Connect__ Method
*__CreatePart__ Method
*__Decode__ Method
*__DeletePart__ Method
*__Descend__ Method
*__Disconnect__ Method
*__Encode__ Method
*__HostDelete__ Method
*__NewMessage__ Method
*__Read__ Method
*__ReadMessage__ Method
*__Write__ Method
*__WriteMessage__ Method

Close

Frequently Asked Questions

General Questions

Abort Method

Flags Property

I don't understand the Flags property. How do I use it?

Write Method

Why doesn't the server accept my command when using the Write method?

Internet Pack - General Questions

Why won't my Internet Pack VBXes load into VB?

Why do I get a GPF when I try to unload my form (or control) from the Done event?

With which TCP/IP stacks have your Internet controls been tested?

How do I enable/disable the Windows 95 Dial-Up Networking connect prompt when my application issues a Connect method?

Why won't my Internet Pack VBX load?

How do I convert my code from BLOCKING (Synchronous) to NON-BLOCKING (Asynchronous)?

How can I detect whether someone has entered an IP or host name?

What is the meaning of Error 20002 "unexpected server response"?

Can you recommend any good books that will help me understand Internet programming better?

Internet Pack - Blocking

I'm unclear on blocking. Can you explain it to me?

Should I use blocking or non-blocking calls?

Why do I keep getting the error "Busy executing asynchronous command"?

Why do I keep getting errors when using an Internet VBX control?

Internet Pack - Debugging

How do I tell what's happening when your control is talking to a server?

How do I retrieve only the headers from a message (TOP functionality)?

Frequently Asked Questions

There is a value for Flags that is used when MailSrcIsHost is also set as a part of the Flags value. The value is MailReadHeaderOnly (= 256). When that is set along with MailSrcIsHost, only the message headers are retrieved. Everything normally associated with an e-mail message will be as it is without the MailReadHeaderOnly flag, except that the message will contain no body of any kind.

How do I send the same message to a list of 100 people?

Frequently Asked Questions

The Mabry Mail control conforms to RFC 821 which allows multiple recipients to be specified. To do this, just create a comma delimited string of valid addresses and assign the entire string to either the To, CC, or BCC property (note -- all recipients are shown in the To and CC property, but the BCC hides the mail list from each user -- see the To, CC, and BCC Properties in the help file for further details).

You could create the string by looping through the contents of a ListBox, or a populated array or Collection object that you create.

Then, when the e-mail is sent, the communication between the client (your app.) and the server allows the client side to first inform the server of all of the intended recipients, and then a single copy of the mail is transmitted to the server, resulting in reduced network traffic.

Why do I get an error that says "Busy executing asynchronous command" when I try to send a message?

Frequently Asked Questions

The Mail control is still performing another command. You need to either key off of the Done event before executing subsequent mail commands, or set Blocking = True. See the Blocking property in the Help file for details of blocking vs. non-blocking modes.

Also, you may be able to utilize the CC or BCC properties if sending the same file to multiple recipients. You can specify multiple recipients in each property by separating the e-mail addresses with commas. The Help file contains details of these two properties.

How do I retrieve the cc: header line from an e-mail message? Can I just read the CC property?

Frequently Asked Questions

The CC property is used for sending a CC to someone. The way that you'd have to retrieve the address(es) of those who were CC'ed on a received mail is by reading through the Headers property. This is a string array (the size is contained in HeadersCount) that contains each header line. Simply run through the Headers property array looking for lines that begin with "CC:".

What is the general procedure for sending mail?

The control is capable of creating multi-part messages and encoding messages. Refer to the ENCODE and MAIL samples that come with the control for encoding/decoding messages.

Here are the basic steps for sending mail:

1. Instruct the control that you want to create a new message by issuing the NewMessage method (or set the Action property to MailActionNewMessage).

2. Set the To property to the e-mail address of the intended recipient (note: multiple recipients can be used by separating the e-mail addresses with commas).

3. Set the CC and BCC properties to valid e-mail addresses if you want to CC or blind CC this e-mail to others (note: multiple recipients can be used by separating the e-mail addresses with commas).

4. Set the Host property.

5. Set the From property (this is the user friendly name of who the is sending the message).

6. Set the EMailAddress property (this is the email address of the sender so the recipient knows to whom to respond).

7. Set the Subject.

8. Set the Headers property as desired. Something like the following is sufficient:

```
Mail1.Headers(Mail1.HeadersCount) = "X-Mailer: Mabry "
```

9. Set the MessageID property. Each mail message must have a "world-unique" ID. Something like the following is sufficient:

```
Mail1.MessageID = Year(Now) & Month(Now) & Day(Now) & Fix(Timer) &
"_MabryMail"
```

10. Set the first element of the Body property, i.e.,

```
Mail1.Body(0) = "blah blah blah"
```

Note: If using the VBX and the message is greater than 32767 bytes, break the message up into several elements of the Body array property.

11. Set the ConnectType property to MailConnectTypeSMTP.

12. Issue the Connect method (or set the Action property to MailActionConnect).

13. Use the Flags property to instruct the control that the source of the message is the Body and the destination is to be the host, i.e.,

```
Mail1.Flags = MailSrcIsBody + MailDstIsHost
```

14. Write the message with the WriteMessage method (or set the Action property to MailActionWriteMessage).

15. Repeat steps 1 through 14 as desired.

What is the general procedure for receiving mail?

The control is capable of handling multi-part messages and decoding messages. Refer to the ENCODE and MAIL samples that come with the control for encoding/decoding messages.

Here are the basic steps for retrieving mail from a host and writing messages to disk:

1. Set the Host property.

2. Set the LogonName property.

3. Set the LogonPassword.

4. Set the ConnectType property to MailConnectTypePOP3.

5. Issue the Connect method (or set the Action property to MailActionConnect).

6. Instruct the control that you want the source of the message to be the host with Flags = MailSrcIsHost

7. If the PopMessageCount property is greater than zero, there is mail to retrieve.

8. Select a message to receive by setting the MessageID property to a valid value (a value between 1 and PopMessageCount).

9. Issue the ReadMessage method (or set the Action property to MailActionReadMessage).

10. Instruct the control that you want the destination of the message to be a file by setting Flags = MailDstIsFile.

11. Determine where you want to store the message by setting the DstFilename property, ie-Mail1.DstFilename = "C:\email\in\mail.msg". Note: if multiple messages are to be received, a naming scheme is needed to prevent previous messages from being overwritten.

12. Write the message with the WriteMessage method (or set the Action property to MailActionWriteMessage).

13. Repeat steps 6 through 12 as desired.

14. Disconnect from the server by issuing the Disconnect method (or set the Action property to MailActionDisconnect).

Do you have a separate control for encoding/decoding?

Frequently Asked Questions

No, we do not have a separate control for encoding/decoding, but the Mail control can be easily adapted for such functionality. Just use the SrcFilename and DstFilename properties and the Decode/Encode methods. These methods are not dependent on any of the "mail-specific" methods like Connect, ReadMessage, etc.

Is there a property that lets you add custom extra fields like x-mailer, Reply-to, Priority and others?

Frequently Asked Questions

Use the EMailAddress property for the Reply-to, then use the Headers property for creating custom headers. The property is a string array, so you would do something like:

```
Mail1.Headers(0) = "X-Mailer: Mabry"
Mail1.Headers(1) = "X-Priority: 3"
```

How do I recover parts (attachments) from forwarded mail? If Mail1.parts was 2 on the original, then, after forwarding, Mail1.parts is 1. As a result, I cannot recover the attachment. What do I do?

The control can handle recursive parts -- when a message has multiple parts embedded within parts, you need to Descend into a Part to access its multiple parts.

In other words, if an original message has two parts and it's forwarded, the new message has one part. If you descend into that one part, the Parts property will be 2 (the two original parts) and you can access them.

If the user cancels Dial-up-Networking, is there a way to reset the Mabry Mail control? Disconnect or Abort will not work because the socket was never connected.

The only thing you can do in that case is wait for the Winsock to timeout and then trap the error.

Alternatively, you could use the latest version of the Mabry RAS control to check to see if the connection exists, and, if not, attempt to establish the connection. If the user aborts the connection, you can prevent the Mail control from attempting to connect by checking the RAS.EntryConnected property before issuing the Mail.Connect method.

Why does the Mabry Mail control strip quotes (") out of e-mail addresses in the To, CC and BCC properties?

Frequently Asked Questions

Quotes should be stripped out. The quote character is a special character and gets stripped out unless preceded by a "\" character.

For example, if you use Netscape to send a message and the From in Netscape is "Joe Blow" (in quotes), the source header gets converted to "\"Joe Blow"\"

You could put the quotes back in, but our tests indicated that some mail clients (i.e., Pegasus) may have problems if the mail is sent with quotes in From.

If I send a message to my normal SMTP server, the message arrives OK. But, if I send the message specifying another SMTP server, I find two boundary lines in the body message. Do you know why?

If you are setting the Host property, the message should be the same regardless of the server -- unless the SMTP server is a re-mailer that forwards the message on to the final destination. If this is the case, the problem is the intermediate mail server (and it's really not a problem -- it is common for forwarded messages).

When this happens, check the Content-Type of the final message. If it is "message" or "message / RFC821", then the intermediate mail server is forwarding (and altering) the message. There is nothing the control can do to correct that.

When I attempt to connect to a server that is unavailable, the operation times out and an "Operation Timed Out" message box pops up. How can I prevent the message box from displaying?

Frequently Asked Questions

Establish the connection asynchronously (Blocking = False).

Why do current messages sent by my program include headers from previous messages?

Frequently Asked Questions

The Headers property will accumulate headers until it is cleared. When creating a new message, use the NewMessage method (or set the Action property to MailActionNewMessage) to clear the Headers property.

How can I shorten the Timeout period during a Connect? My program waits for 2 minutes (in the case of no connection) before returning.

Frequently Asked Questions

The Timeout property is only to be used after connecting to the server (to handle server timeouts vs. WinSock timeouts). WinSock is controlling the connect timeout.

To abort a connect and unload your application if a connection doesn't occur within 15 seconds, set Blocking to False and use a standard Timer control set to 15 seconds to unload the form when it fires. Note -- don't forget to disable the timer if the connect is successful!

I'm using non-blocking (asynchronous) mode, but the Done event never fires for some methods, like NewMessage. Why is this?

Frequently Asked Questions

The Done event only fires when the method invoked involves a call to the winsock interface. Since blocking is a function of the winsock interface, if an action of the control does not involve WinSock, the blocking function is not applicable and the Done event does not fire.

How can I send e-mail when Blocking = False?

Frequently Asked Questions

It is difficult to write non-blocking in an e-mail since code needs to be in various sub-routines. Here is a simple sample of sending in non-blocking mode:

In the Declarations section, define a boolean flag (if using VB3, just make it an integer and change the True/False as necessary):

```
Dim fBusy as Boolean
```

In the Done event, set the fBusy flag equal to False:

```
Private Sub Mail1_Done()
    'this clears the fBusy flag
    fBusy = False
End Sub
```

Then, use the below code to send your mail:

```
    Mail1.Blocking = False
    Mail1.Action = 11 'MailActionNewMessage
    Mail1.To = "somebody@somewhere.com"
    Mail1.From = "My Name"
    Mail1.EMailAddress = "me@mydomain.com"
    Mail1.Subject = "This is the subject"
    Mail1.Body(0) = "This is the message."
    Mail1.Host = "smtp.mydomain.com"
    Mail1.ConnectType = 0 'MailConnectTypeSMTP

    fBusy = True 'set the busy flag -- the Done event will clear it
    Mail1.Action = 3 'MailActionConnect
    Do While (fBusy)
        DoEvents
    Loop

    Mail1.Flags = 64 'MailDstIsHost

    fBusy = True
    Mail1.Action = MailActionWriteMessage
    Do While (fBusy)
        DoEvents
    Loop
    Mail1.Action = MailActionDisconnect
```

Note: you may want to enable a timer to prevent an endless loop in the Do/Loop code.

What is the difference between Buffer and Body properties in the Mail control?

Frequently Asked Questions

The buffer is just a temporary string variable that is useful when encoding/decoding attachments. The actual body of the message is in the Body property, which is a string array (the VBX has a size limitation of 32K -- the OCX does not have the size limitation, so you can just use the first element of the array).

Do you know of any SMTP servers that have known problems with your control?

Frequently Asked Questions

MAIL does not work with a server called IMS Pop3 with when the mail box is empty, but it is clearly the server failing to follow the RFC. The above server does not terminate a multi-line response as specified in the RFC and so the control's method times out.

The RFCs define the protocols and, thus, anyone who creates an application that conforms to the RFC should be able to communicate with anyone else who conforms. The Mabry Mail control conforms with RFC 821 (SMTP) and RFC 1725 (POP3). It is also MIME compatible and complies with RFC 1521 and 1522.

Can the Mabry Mail control be used with MS Exchange?

Frequently Asked Questions

The Mail control complies with POP3 and SMTP protocols. MS Exchange supports these, but Microsoft's Knowledge Base indicates that some versions of Exchange may have problems routing certain SMTP messages. It will accept:

```
techsupport@mabry.com
```

but not

```
Techical Support Staff <techsupport@mabry.com>
```

See http://www.microsoft.com/kb/ for updated information and Exchange Service Pack information.

Can the Mail control be used to write a mail server?

Frequently Asked Questions

No, the Mabry Mail control supports the POP3 and SMTP protocols from the perspective of the client side, not the server side. However, the Mabry ASocket control, which provides direct access to the WinSock interface, could be used as a building block in a mail server.

Can I use the Mail VBX to run multiple instances of my POP3 mail program?

Frequently Asked Questions

The POP3 protocol will not allow two people to access the same mailbox at the same time. Once someone passes through the Authorization state, the mail box is locked and subsequent attempts to access the mailbox will fail until the mailbox is unlocked. This is a function of the mail server.

If trying to run multiple instances accessing different mailboxes, the VBX only runs multiple instances if running in a separate address space (as can be specified under NT) in non-blocking mode. If it is run in the same address space, it will not work.

However, you can run multiple instances using the VBX in VB4-16, or when using the OCX. Note: results may vary depending upon the WinSock used, particularly when using the VBX since 16 bit applications run in a shared address space.

Can I use the Mail VBX to run multiple instances of my SMTP mail program?

Frequently Asked Questions

Using the VBX in VB3, it only runs multiple instances if running in a separate address space (as can be specified under NT). If it is run in the same address space, it will not work.

However, you can run multiple instances using the VBX in VB4-16, or when using the OCX. Note: results may vary depending upon the WinSock used, particularly when using the VBX since 16-bit applications run in a shared address space.

Can the MMAIL32.OCX be used to create an Active Server Object which can be called (no GUI) from an ASP script?

Frequently Asked Questions

Generally, you can't because controls require a container. There is clever work around. Use VB5 to create an ActiveX control and place the Mabry control in the ActiveX control. Using the ActiveX interface wizard, map all the Mabry tool's properties and methods to the ActiveX component.

Then, you should be able to create the control and reference the control using your custom ActiveX control. For example, to use a Mail control on an .ASP page:

```
set myMail = server.createobject("MyProject.MyForm")

myMail.connecttype=0
myMail.host=" ..... "
etc.
```

Note: The Mabry License Agreement explicity states that you cannot use a Mabry control to create an ActiveX control to be marketed as an ActiveX control. You can, however, use the above method to create an ActiveX control to be used in server side web pages or in your applications.

I need to send very large files (3 megabytes) using Microsoft Access. How do I do this?

If you're using the ActiveX (OCX-32) version, it shouldn't be a problem. The Body(0) property can handle the entire encoded file.

If you're using the VBX, you have a string limitation that is found in all VBXes. In this case, you must break the file into smaller segments. You can do this by first encoding it to a temp file (see the Encode example that comes with the control). Then just read in the file in 32K chunks and write each chunk to an element of the Body array. When you decode, you'll need to reverse the process by reading each element of the Body array and writing it to a temp file, then decode.

I have been unable to use the Abort method without getting socket errors. Why?

Frequently Asked Questions

The Abort method is only valid once you've been authenticated by the mail server and are in the Transaction state -- it willl abort transactions but not connections.

I don't understand the Flags property. How do I use it?

When you want to send a message, you really need two flags, one to indicate what is the source of the message and what is the destination. The Flag property in the Help file describe the values and the relevancy of the flags vary depending upon whether you are sending or receiving a message.

In a nutshell, when sending a message the host is usually the destination and the source is usually the Body, a file, or the control's buffer.

Why doesn't the server accept my command when using the Write method?

Frequently Asked Questions

All strings must be terminated with a CR/LF (Chr(13) & Chr(10)). You must add a CRLF at the end of your string.

Why won't my Internet Pack VBXes load into VB?

Frequently Asked Questions

The VBXes are looking for a file called WINSOCK.DLL. This DLL should be in your Windows directory (most DLLs are located in your Windows\System directory -- this one is an exception). Look for WINSOCK.DLL. If it's not in your Windows directory, we recommend moving it there. Be sure to write down where it was, in case something goes wrong.

Also, check the date on your WINSOCK.DLL. If it's 1994 or before, you should look into getting a later version.

Why do I get a GPF when I try to unload my form (or control) from the Done event?

Frequently Asked Questions

This is not uncommon in many controls. If the form containing the control is unloaded but the control's C++ code for the event has to reference the control, the GPF will occur because the control is no longer available after it is has been unloaded. The solution is to enable a timer in the Done event and have the Timer unload the form (or control).

With which TCP/IP stacks have your Internet controls been tested?

Frequently Asked Questions

The majority of our internal testing is done on either NT's or Win95's standard stacks. We also utilize a 3.1 machine running Trumpet Winsock.

As part of our beta program, the controls wind up on a variety of stacks like Novell (known to have differences in Winsock, but should be OK with the latest patches from Novell), WFWG (also has a known problem that can cause FTP trouble, but MS has a patch for that product as well (article ID Q122544)).

The controls support the standard Winsock interface, so in general, the 16-bit environments that do not come with a default stack (I.e., Windows 3.x) may involve a bit more setup, but as long as some reputable stack is used, there shouldn't be any problems.

How do I enable/disable the Windows 95 Dial-Up Networking connect prompt when my application issues a Connect method?

Frequently Asked Questions

The fact that the DUN pops up when attempting to establish a network connection is a Win95 OS setting. To change this behavior, choose Dial Up Networking from "My Computer", and select "Settings..." from the "Connections" menu. Set the desired value in the "When establishing a network connection" frame.

Why won't my Internet Pack VBX load?

Frequently Asked Questions

Usually, the Internet VBXes won't load when the WINSOCK.DLL is missing. Make sure you have a current WINSOCK.DLL in the Windows or Windows\System sub-dir. Some versions of Windows 3.x WinSocks may actually require a TCP/IP connection.

How do I convert my code from BLOCKING (Synchronous) to NON-BLOCKING (Asynchronous)?

Frequently Asked Questions

A quick fix for converting Blocking code to non-blocking code is as follows:

```
Blocking=False
```

In the Declarations of the Form, add:

```
Private fDone as Boolean
```

In the Done event of the control set the fDone flag as shown:

```
Private Sub FTP1_Done()
    fDone = True
End Sub
```

Then, when invoking a method, just loop until the Done event sets the fDone flag.

```
fDone = False
mMail1.Connect
Do
    DoEvents
    'here is where your application
    'can do other things
Loop Until (fDone)
```

Note: you may want to set a timer in the loop so it will not loop endlessly should some problem occur. Also, depending upon your code you may want to conditionally set the fDone flag in the AsyncError event.

How can I detect whether someone has entered an IP or host name?

Frequently Asked Questions

You can use the function (below) to check for a host name or IP address.

```
' This Function receives a string argument and
' validates whether the string is a valid IP value,
' by verifying that it is in the format of w.x.y.z and
' that each octet is between 0 and 255
'
' Returns True if IP there are 4 octets and each is
' between 0 and 255.
'
' Returns False in all other cases
'
' Disclaimer -- this function will not detect certain
' values such as netmasks like 255.255.255.255,
' which meet the criteria but are not valid IPs.
'
Private Function Valid_IP(IP As String) As Boolean
    Dim i As Integer
    Dim dot_count As Integer
    Dim test_octect As String

    IP = Trim$(IP)

    ' make sure the IP long enough before
    ' continuing
    If Len(IP) < 8 Then
        Valid_IP = False
        Exit Function
    End If

    i = 1
    dot_count = 0
    For i = 1 To Len(IP)
        If Mid$(IP, i, 1) = "." Then
            ' increment the dot count and
            ' clear the test octet variable
            dot_count = dot_count + 1
            test_octet = ""
            If i = Len(IP) Then
                ' we've ended with a dot
                ' this is not good
                Valid_IP = False
                Exit Function
            End If
        Else
            test_octet = test_octet & Mid$(IP, i, 1)
            On Error Resume Next
            byte_check = CByte(test_octet)
            If (Err) Then
                ' either the value is not numeric
                ' or exceeds the range of the byte
                ' data type.
                Valid_IP = False
                Exit Function
```

```
            End If
        End If
    Next i
    ' so far, so good
    ' did we get the correct number of dots?
    If dot_count <> 3 Then
        Valid_IP = False
        Exit Function
    End If
    ' we have a valid IP format!
    Valid_IP = True
End Function
```

What is the meaning of Error 20002 "unexpected server response"?

Frequently Asked Questions

The control has issued some command and the server did not accept it. It could be anything from an improperly formatted e-mail address to an unimplemented command on the server. You'll have to enable debugging to see what the command and reply are.

Can you recommend any good books that will help me understand Internet programming better?

Any good book on TCP/IP would be helpful. From personal experience, tech support recommends "TCP/IP" by Dr. Sidnie Feit, published by McGraw-Hill . It is *not* written from a programming standpoint, but does include everything you'd want to know about the lower levels of the OSI stack (including TCP/UDP/IP, etc.).

I'm unclear on blocking. Can you explain it to me?

When your application requests data from a network connection, it is hard to predict how long it will take before the data arrives and the call can complete. As a programmer, you have to determine whether to wait for the outcome of the call, or return immediately to your application and get the data *when* the data arrives.

Calls that wait, are called blocking calls. Because the call must complete before the application continues, blocking calls are also referred to as synchronous calls.

Calls that return control to your application immediately are called non-blocking calls. Since your application can perform tasks while the call is retrieving the data, non-blocking calls are also referred to as asynchronous calls.

Mabry Internet controls support both blocking and non-blocking calls.

It is important to note that even when using blocking calls, Windows can send event messages (such as Timer events, mouse clicks, etc.) to your application and it can respond to them. This can result in errors. It is the responsibility of the programmer to minimize the likelyhood of these situations (such as disabling any Timers or command buttons that will interrupt the call) and handle any errors should such conditions arise.

Error handling is very important when issuing calls to a network. Always use some method of On Error handling when invoking blocking calls. For non-blocking calls, normal On Error handling is required in addition to responding to the AsyncError event.

Should I use blocking or non-blocking calls?

Frequently Asked Questions

It depends on your application. See the explanation on blocking calls for a complete description of blocking vs. non-blocking.

Why do I keep getting the error "Busy executing asynchronous command"?

Frequently Asked Questions

A call has been invoked but a previous call has not been completed yet. Either set Blocking mode to true or wait for the Done event before issuing subsequent commands.

It is important to note that even when using blocking calls, Windows can send event messages (such as Timer events, mouse clicks, etc.) to your application and it can respond to them. This can result in errors. It is the responsibility of the programmer to minimize the likelyhood of these situations (such as disabling any Timers or command buttons that will interrupt the call) and handle any errors should such conditions arise.

Why do I keep getting errors when using an Internet VBX control?

A call has been invoked but a previous call has not been completed yet. Either set Blocking mode to true or wait for the Done event before issuing subsequent commands.

It is important to note that even when using blocking calls, Windows can send event messages (such as Timer events, mouse clicks, etc.) to your application and it can respond to them. This can result in errors. It is the responsibility of the programmer to minimize the likelihood of these situations (such as disabling any Timers or command buttons that will interrupt the call) and handle any errors should such conditions arise.

How do I tell what's happening when your control is talking to a server?

Frequently Asked Questions

The Internet Pack controls have debugging support built-in. Simply set the Debug property on the control to 1 and then add the following code to the Debug event of the control:

```
Debug.Print Message
```

Registration Information

CREDITS

Mail was written by Zane Thomas.

CONTACT INFORMATION

Orders, inquiries, technical support, questions, comments, etc. can be sent to mabry@mabry.com on the Internet. Our mailing address/contact information is:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA 98103-1926

Sales: 1-800-99-MABRY (U.S. Only)
Voice: 206-634-1443
Fax: 206-632-0272 or 206-364-3196
Web: http://www.mabry.com

COST

The price of Mail (control only) is US$40 (US$45 for International orders). The cost of Mail and the C/C++ source code (of the control itself) is US$120 (US$125 for International orders).

Prices are subject to change without notice.

DELIVERY METHODS

We can ship this software to you via air mail and/or e-mail.

Air Mail - you will receive disks, a printed manual, and printed receipt if you choose this delivery method. The costs are:

US$5.00	US Priority Mail
US$10.00	AirBorne Express 2nd Day (US deliveries only)
US$15.00	AirBorne Express Overnight (US deliveries only)
US$45.00	International AirBorne Express.

E-Mail - We can ship this package to you via e-mail. You need to have an e-mail account that can accept large file attachments (which includes CompuServe, AOL, and most Internet providers). If you choose this option, please note: a printed manual is not included. We will, however, e-mail a receipt to you.

Be sure to include your full mailing address with your order. Sometimes (on the Internet) the package cannot be e-mailed, so we are forced to send it through the normal mails.

CompuServe E-Mail - CompuServe members can use the software registration forum (GO SWREG) to register this package. Mail's SWREG ID number is 6395. The source code version's ID number is 9062. PLEASE NOTE: When you order through SWREG, we send the registered package to your CompuServe account (not your Internet or AOL account) within a few hours.

ORDER / PAYMENT METHODS

You can order this software by phone, fax, e-mail, mail. For your convenience, an order form has been provided that you can print out directly from this help file.

Please note that orders must include all information that is requested on our order form. Your shipment WILL BE DELAYED if we have to contact you for additional information (such as phone number, street address, etc.).

You can pay by credit card (VISA, MasterCard, American Express), check (U.S. dollars drawn on a U.S. bank), cash, International Money Order, International Postal Order, Purchase Order (established business entities only - terms net 30), or wire transfer.

WIRE TRANSFER INFORMATION

Here is the information you need regarding our account for a wire funds transfer:

Bank Name:	SeaFirst - Stone Way Branch
Bank Address:	3601 Stone Way North Seattle, WA 98103
Bank Phone:	206-585-4951
Account Name:	Mabry Software, Inc.
Routing Number:	12000024
Account Number:	16311706

If you are paying with a wire transfer of funds, please add US$12.50 to your order. This is the fee that SeaFirst Bank charges Mabry Software. Also, please ADD ANY ADDITIONAL FEES THAT YOUR BANK MAY CHARGE for wire transfer service. If you are paying with a wire transfer, we must have full payment deposited to our account before we can ship your order.

Mail Order Form

Use the Print Topic... command from the File menu to print this order form.

Mail this form to:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA 98103-1926

Phone: 206-634-1443
Fax: 206-632-0272 or 206-364-3196
Internet: mabry@mabry.com
Web: www.mabry.com

Where did you get this copy of Mail?

_____

Name: _____

Ship to: _____

_____

_____

_____

Phone: _____

Fax: _____

E-Mail: _____

MC/VISA/AMEX: _____ exp. _____

P.O. # (if any): _____ Signature _____

qty ordered ____ REGISTRATION
$40.00 ($45.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.

qty ordered ____ SOURCE CODE AND REGISTRATION
$120.00 ($125.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.

Close | Copy | Print

Error Codes

Constant	Value	Description
	0	No error.
WSAEINTR	10004	System level interrupt interrupted socket operation.
WSAEBADF	10009	Generic error for invalid format, bad format.
WSAEACCES	10013	Generic error for access violation.
WSAEFAULT	10014	Generic error for fault.
WSAEINVAL	10022	Generic error for invalid format, entry, etc.
WSAEMFILE	10024	Generic error for file error.
	10025	The IP address provided is not valid or the host specified by the IP does not exist.
WSAEWOULDBLOCK	10035	The socket is marked as non-blocking and the operation would block. You will be notified when the operation completes. This is just a warning, your operation will complete successfully. It is safe to ignore this error code.
WSAEINPROGRESS	10036	This error is returned if any Windows Sockets function is called while a blocking function is in progress.
WSAEALREADY	10037	The asynchronous routine being canceled has already completed.
WSAENOTSOCK	10038	Invalid socket or not connected to remote.
WSAEDESTADDRREQ	10039	A destination address is required.
WSAEMSGSIZE	10040	The socket is of type ASocketDatagram, and the datagram is larger than the maximum supported by the Windows Sockets implementation.
WSAEPROTOTYPE	10041	The specified port is the wrong type for this socket.
WSAENOPROTOOPT	10042	The option is unknown or unsupported.
WSAEPROTONOSUPPORT	10043	The specified port is not supported.
WSAESOCKTNOSUPPORT	10044	The specified socket type is not supported in this address family.
WSAEOPNOTSUPP	10045	The referenced socket is not a type that supports connection-oriented service.
WSAEAFNOSUPPORT	10047	Addresses in the specified family cannot be used with this socket.
WSAEADDRINUSE	10048	The specified address is already in use.
WSAEADDRNOTAVAIL	10049	The specified address is not available.
WSAENETDOWN	10050	The connected network is not available.
WSAENETUNREACH	10051	The connected network is not reachable.
WSAENETRESET	10052	The connected network connection has been reset.
WSAECONNABORTED	10053	The current connection has been aborted by the network or intermediate services.

WSAECONNRESET	10054	The current socket connection has been reset.
WSAENOBUFS	10055	No buffer space is available. The socket cannot be connected.
WSAEISCONN	10056	The socket is already connected.
WSAENOTCONN	10057	The current socket has not been connected.
WSAESHUTDOWN	10058	The connection has been shutdown.
WSAETIMEDOUT	10060	The current connection has timed out.
WSAECONNREFUSED	10061	The requested connection has been refused by the remote host.
WSAENAMETOOLONG	10063	Specified host name is too long.
WSAEHOSTDOWN	10064	Remote host is currently unavailable.
WSAEHOSTUNREACH	10065	Remote host is currently unreachable.
WSASYSNOTREADY	10091	Remote system is not ready.
WSAVERNOTSUPPORTED	10092	Current socket version not supported by application.
WSANOTINITIALISED	10093	Socket API is not initialized.
WSAEDISCON	10101	Socket has been disconnected.
WSAHOST_NOT_FOUND	11001	Remote host could not be found.
WSATRY_AGAIN	11002	Remote host could not be found, try again.
WSANODATA	11004	Remote host could not be found.
	20001	Internal control state error.
	20002	Unexpected server response.
	20003	Login required.
	20004	Login failed.
	20005	Already connected to server.
	20006	Busy executing asynchronous command.
	20007	Can't change blocking mode, busy or connected to server.
	20008	Operation timed out.
	20009	Cannot Ascend, already at top level of message.
	20010	Cannot Descend, message has no parts.
	20011	Invalid user name.
	20012	Invalid password.
	20013	Could not open file.
	20014	Could not close file.

Close Copy Print

Simple Message Send Example

The following example shows how to send a simple message using an SMTP server.

```
    ' Enable blocking mode
    mMail1.Blocking = True

    ' Set header properties
    mMail1.To = "someone@somewhere.com"
    mMail1.From = "Your Name <you@wherever.com>"
    mMail1.Host = "smtphost.wherever.com"
    mMail1.EMailAddress = Chr(34) & "Your Name" & Chr(34) &
"<you@wherever.com.com>"

    ' Put text of message in body
    mMail1.Body(0) = "E-Mail Message"

    ' Connect to server
    mMail1.Action = MailActionConnect

    ' Send e-mail message to server
    mMail1.Flags = MailDstIsHost
    mMail1.Action = MailActionWriteMessage

    ' Disconnect
    mMail1.Action = MailActionDisconnect
```

See Also

Disconnect Method

HostDelete Method

See Also

AsyncError Event

Done Event

See Also

CreatePart Method
Descend Method
Parts Property

See Also

Action Property
Done Event
LastError Property

See Also

CC Property

To Property

See Also

Action Property
AsyncError Event
Done Event
LastError Property

See Also

- **Ascend** Method
- **BodyCount** Property
- **Descend** Method
- **Encode** Method
- **ReadMessage** Method
- **WriteMessage** Method

See Also

Body Property

See Also

Action Property
Decode Method
Encode Method

See Also

BCC Property

To Property

See Also

Abort Method
ConnectType Property
Disconnect Method
Done Event
Host Property
LogonName Property
LogonPassword Property

See Also

Connect Method

Disconnect Method

See Also

ContentDisposition Property

ContentID Property

See Also

ContentSubtype Property

ContentType Property

See Also

Connect Method

MessageID Property

See Also

ContentSubtypeParameters Property

ContentType Property

See Also

ContentSubtype Property

ContentType Property

See Also

ContentType Property
Decode Method
Encode Method

See Also

ContentSubtype Property

ContentSubtypeParameters Property

See Also

Descend Method
HostDelete Method
Part Property
Parts Property

See Also

Connect Method

See Also

Debug Event

See Also

Debug Property

See Also

ContentTransferEncoding Property
DstFilename Property
Encode Method
Flags Property
SrcFilename Property

See Also

Part Property

See Also

Ascend Method

Parts Property

See Also

Abort Method
Connect Method
HostDelete Method

See Also

AsyncError Event

Blocking Property

See Also

- **Decode** Method
- **Encode** Method
- **Flags** Property
- **ReadMessage** Method
- **SrcFilename** Property
- **WriteMessage** Method

See Also

WriteMessage Method

See Also

Decode Method
DstFilename Property
Flags Property
SrcFilename Property

See Also

Decode Method
Encode Method
ReadMessage Method
WriteMessage Method

See Also

BCC Property
CC Property
EMailAddress Property
To Property

See Also

Ascend Method

Descend Method

HeadersCount Property

See Also

Headers Property

See Also

Headers Property

HeadersCount Property

See Also

Action Property
Connect Method
LogonName Property
LogonPassword Property

See Also

Abort Method
Disconnect Method
MessageID Property

See Also

Action Property
AsyncError Event
Done Event

See Also

Headers Property

HeadersCount Property

See Also

Action Property
Connect Method
LogonPassword Property

See Also

Action Property
Connect Method
LogonName Property

See Also

- **Action** Property
- **Ascend** Method
- **ContentSubtype** Property
- **ContentType** Property
- **CreatePart** Method
- **DeletePart** Method
- **Descend** Method
- **Part** Property
- **Parts** Property

See Also

Headers Property

See Also

ContentType Property

See Also

Parts Property

See Also

CreatePart Method
Descend Method
Parts Property

See Also

Part Property

See Also

Connect Method

PopMessageSizes Property

See Also

Connect Method

PopMessageCount Property

See Also

ConnectType Property
Host Property
SmtpPort Property

See Also

Action Property
Done Event
ReadMessage Method
WriteMessage Method

See Also

Done Event
ReadData Property
WriteData Property
Write Method

See Also

Action Property
WriteData Property
Done Event
Read Method

See Also

Action Property
Flags Property
MessageID Property
SrcFilename Property
Timeout Property
WriteMessage Method

See Also

Connect Method
Host Property
PopPort Property

See Also

Action Property
Decode Method
DstFilename Property
Encode Method
Flags Property
ReadMessage Method
WriteMessage Method

See Also

BCC Property
CC Property
ReadMessage Method
To Property
WriteMessage Method

See Also

AsyncError Event

Done Event

See Also

BCC Property
CC Property
ReadMessage Method
WriteMessage Method

See Also

ReadData Property
WriteData Property
Done Event
Read Method

See Also

Action Property
ReadData Property
Write Method

See Also

Action Property
Buffer Property
DstFilename Property
Flags Property
ReadMessage Method
Timeout Property

Abort Method

See Also Error Codes Frequently Asked Questions

Description

Aborts the current POP3 connection. Not valid for SMTP connections.

Syntax

*object.***Abort**

The syntax of the **Abort** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

When you connect to a POP3 server, a transaction is started. During the transaction you may delete messages using the HostDelete method. A transaction is terminated by using either the Disconnect or Abort methods. Using Abort terminates the current transaction and begins another transaction (it does not disconnect from the server). When a transaction is terminated this way all, messages deleted with the HostDelete method are not deleted by the server.

AboutBox Method

Error Codes Frequently Asked Questions

Description

displays the About Box for the control.

Syntax

*object.***AboutBox**

The syntax of the **AboutBox** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

This method displays the About Box for this control to display copyright information.

Action Property

See Also Error Codes Frequently Asked Questions

Description

Causes control to initiate a command / method.

Syntax

*object.***Action** [= *integer*]

The syntax of the **Action** property has these parts:

Part	Description
object	A Mail control.
integer	An integer that specifies the action to perform.

Remarks

Setting this property makes the Mail control perform an action. The action depends on the value set. Mail accepts the following:

Constant	Value	Description
MailActionNoAction	0	No action
MailActionAbort	1	Abort POP3 Transaction (Abort)
MailActionAscend	2	Ascend out of message part (Ascend)
MailActionConnect	3	Connect to host system (Connect)
MailActionCreatePart	4	Create a new message part (CreatePart)
MailActionDecode	5	Decode data (Decode)
MailActionDeletePart	6	Delete current message part (DeletePart)
MailActionDescend	7	Descend into current message part (Descend)
MailActionDisconnect	8	Disconnect from host system (Disconnect)
MailActionEncode	9	Encode data (Encode)
MailActionHostDelete	10	Delete received message from POP3 server (HostDelete)
MailActionNewMessage	11	Create new message (NewMessage)
MailActionRead	12	Read data from server (Read)
MailActionReadMessage	13	Read a message (ReadMessage)
MailActionWrite	14	Write data to a server (Write)
MailActionWriteMessage	15	Write a message (WriteMessage)

Data Type

Integer

Ascend Method

See Also Error Codes Frequently Asked Questions

Description

Ascends out of the current message part.

Syntax

*object.***Ascend**

The syntax of the **Ascend** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

Ascends from the current message up to the previous message in a multi-part message. All properties are set to the current message after the ascend operation is completed.

When Ascend is executed, all relevant properties will be saved and associated with the message part from which you are ascending. To create parts (attachments) use the CreatePart method to create the new part. Descend into the part with Descend. Assign message contents and properties as required and then Ascend out of the part.

AsyncError Event

See Also Error Codes Frequently Asked Questions

Description

Fired when an error occurs during asynchronous operations

Syntax

Sub *object*_**AsyncError(**[*index* **As Integer**,] *errornumber* **As Integer**, *errormessage* **As String)**

The syntax of the **AsyncError** event has these parts:

Part	Description
object	A Mail control.
index	An integer that identifies a control if it's in a control array.
errornumber	An integer that holds the current error number.
errormessage	A string expression that holds text describing the current error.

Remarks

If an error occurs during the execution of asynchronous commands (only possible when Blocking is set to False) the program is notified by firing the AsyncError event.

BCC Property

See Also Error Codes Frequently Asked Questions

Description

Blind Carbon Copy recipients of current message.

Syntax

*object.***BCC** [= *string*]

The syntax of the **BCC** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that contains a list of e-mail addresses.

Remarks

Three properties are used to specify the recipients of any given e-mail message, the To, CC, and BCC properties. All recipients will see the recipients in the To and CC properties in the header of the copy of the e-mail message which they receive. Recipients listed in the BCC field will not be known to other recipients.

Multiple recipients may be specified and, if so, they must be separated by commas. The following are examples of valid addresses:

```
zane@mabry.com
<zane@mabry.com>
Zane Thomas <zane@mabry.com>
```

If you choose to include a full name as well as an e-mail address, as is done in the third example, you must enclose the actual e-mail address in angle brackets.

You should not use other formats for specifying an e-mail address. The Mail control will attempt to extract the actual e-mail address from the string specified. You may find, however, that the Mail control does not recognize them.

Data Type

String

Blocking Property

See Also Error Codes Frequently Asked Questions

Description

Determines if any methods or actions are blocking.

Syntax

*object.***Blocking** [= *boolean*]

The syntax of the **Blocking** property has these parts:

Part	Description
object	A Mail control.
boolean	A boolean flag that determines if the control waits until a command is finished, or, returns control immediately and then fires an event when done.

Remarks

If this property is set to True, any commands using either the Action property or any of the methods will not return to your code until the command completes. In other words, the command will be handled synchronously.

If this property is false, any commands are handled asynchronously. They return to you immediately. You are notified of completion with the Done event. When the user-interface requires that the user be able to cancel file transfers, it is recommended that Blocking be set to False so that your program can respond quickly to user actions.

Important Note: The Blocking property must not be changed while a connection is established. You should only change the Blocking property when the control is not connected to a server. Changing the Blocking property while connected could cause some strange behavior in the control.

Data Type

Integer (boolean)

Body Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

An array containing the body of an e-mail message.

Syntax

*object.***Body(** *index* **)** [= *body*]

The syntax of the **Body** property has these parts:

Part	Description
object	A Mail control.
index	An integer that identifies a part of the Body array.
body	A variant that specifies the text for the body piece identified by *index*.

Remarks

The Body property is an array, so that when you're using the VBX version, you will have a way to access the bodies of very large e-mail messages. Received messages which are greater than just under 32K will result in at least two elements in the array when you're using the VBX. The OCX version of Mail puts the entire body of received e-mail messages in element number zero of the array.

When using the VBX to send e-mail with a large body, you must store the body in pieces with the pieces assigned to elements numbered from zero through the number of elements required to store the entire body.

When using the OCX, there is no practical restriction on the number of bytes you assign to the first element of the array.

The BodyCount property lets you know how many elements are in the Body array. Code which is written so that it would work with a VBX by taking note of the value stored in the BodyCount property will work without change in the OCX.

Note that the data type for this property is Variant so that you may store strings containing null (&h00) characters in the body of a message. This would usually be followed by encoding the body contents into Base64 (or one of the other supported formats) prior to transmission.

Also note that any message or message part containing a multi-part message will have no body. The body of a multi-part message is the parts which must be accessed using the Descend method.

Data Type

Variant

BodyCount Property

See Also Error Codes Frequently Asked Questions

Description

The number of elements in the Body property array.

Syntax

*object.***BodyCount**

The syntax of the **BodyCount** property has these parts:

Part	Description
object	A Mail control.

Remarks

Some e-mail messages have text in their bodies which contains more bytes than can be returned as the value of a string by the VBX version of the Mail control. For that reason the Body property is an array and the BodyCount property lets you know how many elements are in the array.

This property is read-only and only available at run-time.

Data Type

Integer

Buffer Property

See Also Error Codes Frequently Asked Questions

Description

A general-purpose buffer.

Syntax

*object.***Buffer** [= *string*]

The syntax of the **Buffer** property has these parts:

Part	Description
object	A Mail control.
string	A string expression.

Remarks

A few methods, such as Encode and Decode, require both a source and destination. The Buffer property can be used as either the source or the destination under some circumstances. Note that the utility of the Buffer property is somewhat limited in the VBX due to the limitation on the length of the string which it may contain. However, since the Buffer property is provided merely as a convenience and all Mail related tasks can be accomplished without it, the VBX limitation is not significant.

Data Type

String

CC Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

List of message recipients who should receive copies of the e-mail message.

Syntax

*object.***CC** [= *string*]

The syntax of the **CC** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that contains a list of e-mail addresses.

Remarks

Three properties are used to specify the recipients of any given e-mail message, the To, CC, and BCC properties. All recipients will see the recipients in the To and CC properties in the header of the copy of the e-mail message which they receive. Recipients listed in the BCC field will not be known to other recipients.

Multiple recipients may be specified and, if so, they must be separated by commas. The following are examples of valid addresses:

```
zane@mabry.com
<zane@mabry.com>
Zane Thomas <zane@mabry.com>
```

If you choose to include a full name as well as an e-mail address, as is done in the third example, you must enclose the actual e-mail address in angle brackets.

You should not use other formats for specifying an e-mail address. The Mail control will attempt to extract the actual e-mail address from the string specified. You may find, however, that the Mail control does not recognize them.

Data Type

String

Connect Method

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

Connects to a mail server.

Syntax

*object.***Connect**

The syntax of the **Connect** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

Connects to either an SMTP or POP3 server, depending upon the setting of the ConnectType property. Before executing the Connect method you must first set the Host property. When connecting to a POP3 server you must also set the LogonName and LogonPassword properties.

Important Note: The Blocking property must not be changed while a connection is established. You should only change the Blocking property when the control is not connected to a server. Changing the Blocking property while connected could cause some strange behavior in the control.

ConnectType Property

See Also Error Codes Frequently Asked Questions

Description

Specifies type of server to use.

Syntax

*object.***ConnectType** [= *integer*]

The syntax of the **ConnectType** property has these parts:

Part	Description
object	A Mail control.
integer	An integer that specifies the type of connection to use.

Remarks

The Mail control supports POP3 and SMTP servers. Before using the Connect method to connect to a server, you should set the ConnectType property as shown in the table below. After executing the Disconnect method, you can change the ConnectType property if required.

Constant	Value	Description
MailConnectTypeSMTP	0	Connect to SMTP server
MailConnectTypePOP3	1	Connect to POP3 server

Data Type

Integer (enumerated)

ContentDescription Property

Description

User-friendly description of an e-mail message.

Syntax

*object.***ContentDescription**

The syntax of the **ContentDescription** property has these parts:

Part	Description
object	A Mail control.

Remarks

A user-friendly description of the content of an e-mail message (or part of a multi-part message) may be provided. For instance, you could assign "This is an image in Gif89a format" as the value of this property when e-mailing a GIF file.

Data Type

String

ContentDisposition Property

See Also Error Codes Frequently Asked Questions

Description

Provides a recommended disposition to the message recipient.

Syntax

*object.***ContentDisposition** [= *string*]

The syntax of the **ContentDisposition** property has these parts:

Part	Description
object	A Mail control.
string	A string that contains the recommended disposition for an e-mail message (or part, in a multi-part message).

Remarks

MIME messages may have a ContentDisposition header. ContentDisposition is used to indicate two things: whether a message part should be displayed as an attachment or inline, and the suggested name of a file for storing the message.

Examples:

`ContentDisposition = "attachment; filename=foo.zip"` indicates that the contents should be displayed as an attachment and the suggested filename for storing the contents is foo.zip.

`ContentDisposition = "inline; filename=me.gif"` indicates that the contents should be displayed inline and that the suggested filename is me.gif.

Note that you would expect to find the first example only in a body part which is application/zip or application/x-zip-compressed while the second example might appear in a body part which is image/gif.

Data Type

String

ContentID Property

See Also Error Codes Frequently Asked Questions

Description

An arbitrary unique content identifier.

Syntax

*object.***ContentID** [= *string*]

The syntax of the **ContentID** property has these parts:

Part	Description
object	A Mail control.
string	A string that specifies a unique content identifier.

Remarks

In constructing a high-level user agent, it may be desirable to allow one body to make reference to another. Accordingly, bodies may be labeled using the ContentID property. Note that ContentID has special meaning in the case of the multi-part/alternative content type. You should refer to RFC1521 for the details of constructing such messages.

Data Type

String

ContentSubtype Property

See Also Error Codes Frequently Asked Questions

Description

Specifies a message's subtype.

Syntax

*object.***ContentSubtype** [= *string*]

The syntax of the **ContentSubtype** property has these parts:

Part	Description
object	A Mail control.
string	A string that specifies the sub-type for a message or message part (in a multi-part message).

Remarks

Each part of a MIME message has a type and subtype. The ContentSubtype property specifies the subtype corresponding to a particular message part. The following table lists the most common types and subtypes.

Type	Subtype	Description
text	plain	Text/plain is the default format for Internet mail messages.
multi-part	mixed	Used for independent body parts which do not need to be bundled in any particular order. Unrecognized multi-part SubTypes are treated as mixed as specified by RFC1521.
	alternative	Used when alternative views of the same message are bundled in a multi-part message. For instance, a multi-part/alternative message might contain a plain text version and a rich text version of the same message body.
	parallel	A message whose parts are intended to be displayed simultaneously. For instance, a parallel message might contain an audio part and a text part. In that case the intent of the sender is that the audio is played while the text is displayed.
	digest	Similar to the multi-part/mixed type except that the bundled messages default to type message/rfc822 instead of text/plain. The control makes this detail largely irrelevant but for completeness, multi-part/digest messages are indicated as being so.
Message	rfc822	Indicates that the body of the message is an encapsulated message. Unlike top-level RFC 822 messages, it is not required that each message/rfc822 body must include a "From", "Subject", and at least one destination header. Also note that the encapsulated message may itself be a MIME message and it is necessary to use the Descend method to access the headers and body of the encapsulated message.
	partial	Mail transports or mail programs may arbitrarily

		break large messages into multiple partial messages. The application (the program using this control) must locate the pieces of a partial message, reassemble them into a complete message, and then use either the MessageFromFile or MessageFromBuffer methods to create a complete (decoded if necessary) message.
	external-body	Indicates that the body of the message is stored external to the message. The ContentsSubtypeParameters property provides further information on how to retrieve the message body.
application	octet-stream	Indicates that the body contains binary data.
	zip	Indicates that the body contains a ZIP file
	x-zip-compressed	Indicates that the body contains a ZIP file
	x-uuencode	A wrapped UUencoded message
	mac-binhex40	A BinHex message
	x-*	Any other subtype of application beginning with x-.
image	jpeg	The message body is a JPEG graphic.
	gif	The message body is a GIF graphic.
audio	basic	The message body is 8bit ISDN mu-law [PCM] data, a single channel recorded at 8000hz.
video	mpeg	The message body is an MPEG animation.

Data Type

String

ContentSubtypeParameters Property

See Also Error Codes Frequently Asked Questions

Description

Specifies parameters used with various ContentSubtypes.

Syntax

*object.***ContentSubtypeParameters** [= *string*]

The syntax of the **ContentSubtypeParameters** property has these parts:

Part	Description
object	A Mail control.
string	A string that specifies the sub-type parameters for a message or message part (in a multi-part message).

Remarks

Many of the ContentType/ContentSubtype combinations also require one or more parameters. The following table lists the parameters specified in RFC1521.

Type / Subtype	Parameter	Description
text/plain	charset	Specifies the character set which should be used when displaying the message.
multi-part/*	boundary	All of the parts of a multi-part message are separated by the character string given as the value of the boundary parameter. Usually you will not need to parse the individual parts of a multi-part message since the control does it automatically. If you need to parse the message yourself, you should read RFC1521 for the details on using the boundary parameter.
message/partial	number	Partial message parts are numbered 1 through the total number of parts. While it is recommended that all partial messages give both the current number and total number parameters, RFC1521 only requires the last part of a set of partial messages to specify the total number.
	total	Total number of parts in the partial message. Required only for the last part but may be present in other parts.
	id	A "world unique" identifier used to associate the parts of a partial message with each other.
Message/external-body	access-type	Access-type and the other message/external-body parameters have numerous uses. You should read RFC1521 for a complete understanding of these parameters.
	name	
	expiration	
	size	

	permission	
	site	
	directory	
	mode	
	server	
	subject	
application/octet-stream	type	General type of data, intended for user viewing and not for automatic processing.
	padding	Number of bits of padding appended. Used only for bit-stream data.
application/zip	name	Recommended filename for zip file.

Data Type

String

ContentTransferEncoding Property

See Also Error Codes Frequently Asked Questions

Description

Describes how the message data was (or should be) encoded prior to transmission.

Syntax

*object.***ContentTransferEncoding** [= *string*]

The syntax of the **ContentTransferEncoding** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies an encoding method.

Remarks

The original SMTP specification, RFC821, restricts mail messages to 7-bit US-ASCII data formatted into lines of no more than 1000 characters. Many of the content types may be used to send data which is not 7-bit ASCII and which may not reasonably be broken up into lines of no more than 1000 characters. To provide for the transmission of such data, the MIME standard (RFC1521) defines the following transfer encoding types:

```
quoted-printable
base64
8bit
binary
7bit
```

Of these standard types, only quoted-printable and Base64 indicate that the message body contains data which has actually been encoded. The other types (8bit, binary, and 7bit) simply indicate the type of data contained in the message.

The MIME standard also allows extensions to the ContentTransferEncoding header. Such extensions are specified with the prefix 'x-'. For instance, x-uuencode is often used to indicate UUencoding. Message parts with ContentTransferEncoding x-uuencode and mac-binhex40 are sometimes received. The mail control supports decoding and encoding message parts using these formats in addition to the standard MIME encoding formats listed above.

The contents of the ContentTransferEncoding property are used by the Encode and Decode methods to determine how to encode/decode data. For this purpose, the following are the only valid values for ContentTransferEncoding:

```
Base64
quoted-printable
mac-binhex40
x-uuencode
```

You can extend the functionality of your e-mail client by recognizing non-standard ContentTransferEncoding values in received messages (such as x-binhex) and modifying the value of the property appropriately before invoking the Decode method.

Data Type

String

ContentType Property

See Also Error Codes Frequently Asked Questions

Description

Describes the data contained in a message or message part.

Syntax

*object.***ContentType** [= *string*]

The syntax of the **ContentType** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the content type for a message or message part.

Remarks

ContentType categorizes the contents of a message into one of a number of broad groups. The specific type of data in a message is further specified by the ContentSubtype and ContentSubtypeParameters properties. It is possible that a message will be received which is of a type not described below. Any such messages are non-standard and if not specific to your e-mail implementation, you can probably do no better than to simply allow the user to save the contents to a file.

Type	Description
text	Simple text message containing textual information in one of a number of character sets (as specified by the ContentSubtypeParameters property).
Multi-part	Message is a multi-part MIME message. Used to combine several body parts, possibly of differing types of data, into a single message.
application	Application specific message.
message	Encapsulates another mail message, most often seen in forwarded messages.
image	Still images such as gif and jpeg.
audio	Audio or voice data.
video	Video or moving image data.

Data Type

String

CreatePart Method

See Also Error Codes Frequently Asked Questions

Description

Creates a new part in a MIME multi-part message.

Syntax

*object.***CreatePart**

The syntax of the **CreatePart** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

The new part is created with the following defaults:

```
ContentDescription  = ""
ContentID = ""
ContentSubtype = "plain"
ContentSubtypeParameters = "charset=us-ascii"
ContentType = "text"
```

After the new part is created, the Parts property will be one greater than it was before CreatePart was executed, and Part will be set to select the newly created part. Before adding content to the new part or modifying any of its headers, you must use Descend to descend into the new part.

Date Property

See Also Error Codes Frequently Asked Questions

Description

Date message sent.

Syntax

*object.***Date** [= *string*]

The syntax of the **Date** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the date and time a message was created.

Remarks

The date and time an e-mail message was sent in standard RFC822 format:

```
Wed, 30 Oct 1996 00:58:56 -0500
```

The -0500 indicates that the local time (00:58:56) at the locale from which the message was sent is 5 hours earlier than Greenwich Mean Time (GMT).

Data Type

String

Debug Property

See Also Error Codes Frequently Asked Questions

Description

Enables and disables the Debug event.

Syntax

*object.***Debug** [= *debugflag*]

The syntax of the **Debug** property has these parts:

Part	Description
object	A Mail control.
debugflag	An integer that determines if Debug events are fired.

Remarks

Setting Debug to one (1) enables the Debug event. Setting it to zero (0) disables the Debug event. All other values are invalid at this time.

Data Type

Integer (enumerated)

Debug Event

See Also Error Codes Frequently Asked Questions

Description

Fired when the control has debugging information for the program.

Syntax

Sub *object*_**Debug(**[*index* **As Integer**,] *message* **As String)**

The syntax of the **Debug** event has these parts:

Part	Description
object	A Mail control.
index	An integer that identifies a control if it's in a control array.
message	A string expression that holds a debugging message from the control.

Remarks

The Debug event is enabled by setting the Debug property to a non-zero value (1 is the only permitted non-zero value at this time). When the Debug property is non-zero, the Debug event will fire as messages are sent to and received from the server. Printing the Message argument string to Visual Basic's debug window will help you understand what is happening as you debug your application.

Decode Method

See Also Error Codes Frequently Asked Questions

Description

Decodes data encoded with Base64, BinHex40, quoted-printable or UUencoding.

Syntax

*object.***Decode** *encoding*, *flags*, *source*, *destination*

The syntax of the **Decode** method has these parts:

Part	Description
object	Required. A Mail control.
encoding	Optional. A string expression that specifies the encoding format.
flags	Optional. An integer expression that specifies the source and destination locations.
source	Optional. A string expression that holds the source data (if *flags* includes MailSrcIsBuffer).
destination	Optional. A string expression that receives the decoded data, if the *flags* parameter includes MailDstIsBuffer.

Remarks

When using the VBX version of Mail, the encoding format is specified by the ContentTransferEncoding property. When using the Decode method with the OCX version, you may override the ContentTransferEncoding property property by supplying a value for the encoding parameter. Acceptable values for the ContentTransferEncoding property property (or the encoding parameter) are:

```
base64
binhex40
quoted-printable
x-uuencode
```

The Flags property (*flags* parameter) specifies the source for data to decode and its destination. The following flag values are acceptable:

Flag	Meaning
MailSrcIsBuffer	Decode contents of Buffer
MailSrcIsFile	Decode contents of file named by the SrcFilename property
MailSrcIsBody	Decode contents of the current message part's body
MailDstIsBuffer	Decoded contents are placed in the Buffer property
MailDstIsFile	Decoded contents are written to file named by the DstFilename property
MailDstIsBody	Decoded contents are written to the current message part's body

When using the OCX version, buffers and/or filenames can be used as the optional source and destination parameters. The Flags property determines how these optional parameters are used.

DeletePart Method

See Also Error Codes Frequently Asked Questions

Description

Deletes the current part.

Syntax

*object.***DeletePart**

The syntax of the **DeletePart** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

Deletes the part currently selected by the Part property. This can be useful if the user decides to delete an attachment.

Descend Method

See Also Error Codes Frequently Asked Questions

Description

Descends into the current message part.

Syntax

*object.***Descend**

The syntax of the **Descend** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

When a message has multiple MIME parts, the Parts property will be greater than zero. When this is the case, a part can be selected by using the Descend method. After the method executes, the control's properties will be set to reflect the contents of the part into which you descended.

Disconnect Method

Description

Performs a normal disconnect from the mail server.

Syntax

*object.***Disconnect**

The syntax of the **Disconnect** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

When connected to an SMTP server the Disconnect method simply disconnects. Disconnect has special semantics when you're connected to a POP3 server. Messages which are marked for deletion using the HostDelete method are deleted from the POP3 user's mailbox by the server upon the successful completion of the Disconnect method.

Important Note: The Blocking property must not be changed while a connection is established. You should only change the Blocking property when the control is not connected to a server. Changing the Blocking property while connected could cause some strange behavior in the control.

Done Event

See Also Error Codes Frequently Asked Questions

Description

This event procedure is fired when the control completes an action or method.

Syntax

Sub *object*_**Done(**[*index* **As Integer**]**)**

The syntax of the **Done** event has these parts:

Part	Description
object	A Mail control.
index	An integer that identifies a control if it's in a control array.

Remarks

Fired when a method has finished executing without error. If an error occurs during execution of any method and Blocking is True, then an error is thrown by the control and must be handled by **On Error**.

When Blocking is False, an error may be thrown during execution of a method as it is when Blocking is True. However, errors which occur during background execution of methods when Blocking is False will result in firing of the AsyncError event.

DstFilename Property

See Also Error Codes Frequently Asked Questions

Description

The destination filename for certain methods.

Syntax

*object.***DstFilename** [= *filename*]

The syntax of the **DstFilename** property has these parts:

Part	Description
object	A Mail control.
filename	A string expression that specifies a filename.

Remarks

Some of the methods, such as Encode and Decode, may use a destination file. When that is the case, the destination filename is given by the DstFilename property.

Data Type

String

EMailAddress Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

E-mail address of the message sender.

Syntax

*object.***EMailAddress** [= *address*]

The syntax of the **EMailAddress** property has these parts:

Part	Description
object	A Mail control.
address	A string expression that specifies the e-mail address of the message sender.

Remarks

SMTP servers like to know who is sending e-mail. The EMailAddress Property is used to set the e-mail address given to the SMTP server prior to sending mail. The simplest form of the required e-mail address is:

```
zane@mabry.com
```

However, you should provide a user-friendly version as in the following:

```
"Zane Thomas" <zane@mabry.com>
```

Data Type

String

Encode Method

See Also Error Codes Frequently Asked Questions

Description

Encodes data using either Base64, quoted-printable, BinHex40, or UUencoding.

Syntax

*object.***Encode** *encoding*, *flags*, *source*, *destination*

The syntax of the **Encode** method has these parts:

Part	Description
object	Required. A Mail control.
encoding	Optional. A string expression that specifies the encoding format.
flags	Optional. An integer expression that specifies the source and destination locations.
source	Optional. A string expression that holds the source data (if *flags* includes MailSrcIsBuffer).
destination	Optional. A string expression that receives the encoded data, if the *flags* parameter includes MailDstIsBuffer.

Remarks

When using the VBX version of Mail, the encoding format is specified by the ContentTransferEncoding property. When using the Encode method with the OCX version, you may override the ContentTransferEncoding property by supplying a value for the encoding parameter. Acceptable values for the ContentTransferEncoding property (or the encoding parameter) are:

```
base64
binhex40
quoted-printable
x-uuencode
```

The Flags property (*flags* parameter) specifies the source for data to encode and its destination. The following flag values are acceptable:

Flag	Meaning
MailSrcIsBuffer	Encode contents of Buffer
MailSrcIsFile	Encode contents of file named by the SrcFilename property
MailSrcIsBody	Encode contents of the current message part's body
MailDstIsBuffer	Encoded contents are placed in the Buffer property
MailDstIsFile	Encoded contents are written to file named by the DstFilename property
MailDstIsBody	Encoded contents are written to the current message part's body

When using the OCX version, buffers and/or filenames can be used as the optional source and destination parameters. The Flags property determines how these optional parameters are used.

Flags Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

Used to specify options for some methods.

Syntax

*object.***Flags** [= *flags*]

The syntax of the **Flags** property has these parts:

Part	Description
object	A Mail control.
flags	A long integer that specifies options.

Remarks

The Encode, Decode, WriteMessage, and ReadMessage methods all require both a source and a destination. The Flags property is used to select the source and destination for those methods. The following table lists the possible values which may used:

Constant	Value	Description
MailSrcIsBuffer	1	Source is Buffer.
MailSrcIsFile	2	Source is file named by SrcFilename property.
MailSrcIsHost	4	Source is mail host
MailSrcIsBody	8	Encode/decode source is message body
MailDstIsBuffer	16	Destination is Buffer.
MailDstIsFile	32	Destination is file named by DstFilename property.
MailDstIsHost	64	Destination is mail host
MailDstIsBody	128	Encode/decode destination is message body
MailReadHeaderOnly	256	Only download the message headers from the POP server (useful for scanning mail). When this flag is set, only the headers are retrieved. Everything normally associated with an e-mail message will be as it is without the MailReadHeaderOnly flag, except that the message will contain no body text of any kind.

All uses of the Flags property require that values from the table be OR'ed together. One value specifies the source data for a method and the other specifies the destination. For instance, if the ReadMessage method is used, an e-mail message can be retrieved from the host and stored in a file as shown in the following code fragment:

```
Mail1.Flags = MailSrcIsHost Or MailDstIsFile
Mail1.DstFilename = "c:\foo"
Mail1.Action = MailActionReadMessage
```

Data Type

Integer (long)

From Property

Description

The sender of an e-mail message.

Syntax

*object.***From** [= *string*]

The syntax of the **From** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the data for the "From:" header line.

Remarks

Normal e-mail messages will contain a "From:" header which contains the e-mail address of the message sender. However, email advertisers and other spam-o-gram generators often place invalid e-mail addresses in this header field.

Data Type

String

Headers Property

See Also Error Codes Frequently Asked Questions

Description

Contents of an e-mail message's header lines.

Syntax

*object.***Headers(** *index* **)** [= *string*]

The syntax of the **Headers** property has these parts:

Part	Description
object	A Mail control.
index	An integer that identifies a particular header line.
string	A string expression that specifies the name and data for a header line.

Remarks

When receiving messages, the Mail control parses the message headers and assigns the common headers (such as To, From, and Subject) to their respective properties. In addition to this, all headers are placed in the Headers array, one header in each element of the array. The HeadersCount property lets you know how many headers are in the array at any time.

Note: As with all other message-related properties, the Headers array changes when you Descend into, or Ascend out of, part of a multi-part message.

Data Type

String

HeadersCount Property

See Also Error Codes Frequently Asked Questions

Description

Number of elements in the Headers property array.

Syntax

*object.***HeadersCount** [= *count*]

The syntax of the **HeadersCount** property has these parts:

Part	Description
object	A Mail control.
count	A long integer that specifies the number of header line entries.

Remarks

This property lets you know how many headers were received for a message (or the current part of a multi-part message). The headers are stored in the Headers property.

Data Type

Integer (long>

HeaderText Property

Description

Contains the entire header for a message.

Syntax

*object.***HeaderText** [= *string*]

The syntax of the **HeaderText** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the entire set of headers for a message or message part.

Remarks

You may access all of the headers for a message at once by using the HeaderText property. This is functionally equivalent to iterating over the Headers array and concatenating all of the elements of that array.

Data Type

String

Host Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

Name or IP address of the SMTP or POP3 host.

Syntax

*object.***Host** [= *string*]

The syntax of the **Host** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the name or IP address of the host.

Remarks

Used to specify the name (or IP address directly) of the SMTP or POP3 host to use during execution of the next Connect method.

Data Type

String

HostDelete Method

See Also Error Codes Frequently Asked Questions

Description

Deletes a message from the POP3 mailbox (on the server/host).

Syntax

*object.***HostDelete**

The syntax of the **HostDelete** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

When you connect to a POP3 server a transaction is started. During the transaction you may mark messages for deletion by storing the message ID in the MessageID property and invoking the HostDelete method.

There are two ways to terminate a POP3 connection, Abort and Disconnect. Abort terminates the current transaction without actually deleting any messages and starts a new transaction. Disconnect performs a normal disconnect and any messages marked for deletion by the HostDelete command will be deleted from the POP3 user's mailbox.

LastError Property

See Also Error Codes Frequently Asked Questions

Description

Holds the last error number reported.

Syntax

*object.***LastError**

The syntax of the **LastError** property has these parts:

Part	Description
object	A Mail control.

Remarks

This property contains the result of the last method executed. It is zero if the last method completed without error. Otherwise, it contains an error code.

This property is read-only and only available at run-time.

Data Type

Integer

Lines Property

See Also Error Codes Frequently Asked Questions

Description

Number of lines in the current e-mail message.

Syntax

*object.***Lines** [= *long*]

The syntax of the **Lines** property has these parts:

Part	Description
object	A Mail control.
long	A long integer that specifies the number of lines of text in the message body.

Remarks

Some e-mail programs provide a "Lines:" header line to let you know how many lines of text are contained in the current e-mail message.

Data Type

Integer (long)

LogonName Property

See Also Error Codes Frequently Asked Questions

Description

User name on the server.

Syntax

*object.***LogonName** [= *string*]

The syntax of the **LogonName** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the user name to use when connecting.

Remarks

Most POP3 hosts require that connecting clients be known to them and so require a logon name and password. The LogonName and LogonPassword properties must be set before attempting to connect to a POP3 host which requires authentication.

Data Type

String

LogonPassword Property

See Also Error Codes Frequently Asked Questions

Description

Host logon password.

Syntax

*object.***LogonPassword** [= *string*]

The syntax of the **LogonPassword** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the password to use when connecting.

Remarks

Most POP3 hosts require that connecting clients be known to them and so require a logon name and password. The LogonName and LogonPassword properties must be set before attempting to connect to a POP3 host which requires authentication.

Data Type

String

MessageClass Property

See Also Error Codes Frequently Asked Questions

Description

The e-mail message type.

Syntax

*object.***MessageClass** [= *string*]

The syntax of the **MessageClass** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the general classification of the current message.

Remarks

Defines the general classification of an e-mail message.

Value	Description
Generic	A generic e-mail message which is not identified as belonging to any other MessageClass category.
MIMEVersion 1.0	A MIME version 1.0 e-mail message which may have multiple parts. The ContentType and ContentSubtype properties further identify the contents of a MIME e-mail message. Parts will be set appropriately for the current message. See the Part and Parts properties and the Ascend, Descend, CreatePart and DeletePart methods for further discussion of multi-part message handling.

Data Type

String

MessageID Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

An e-mail message's "world unique" identification.

Syntax

*object.***MessageID** [= *string*]

The syntax of the **MessageID** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that uniquely identifies a message.

Remarks

RFC821 requires that all e-mail messages have a "world unique" message ID. How exactly one can guarantee that a unique message ID is generated is not specified. However, a reasonably long MessageID composed of the user's name and something like TimeSerial should work under most circumstances.

Note: Most SMTP servers will generate a MessageID header line for you if you do not supply one. You, however, should not rely on this.

Data Type

String

MultipartBoundary Property

Description

Message-unique string which separates message parts.

Syntax

*object.***MultipartBoundary** [= *string*]

The syntax of the **MultipartBoundary** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the string used to separate parts in a multi-part message.

Remarks

The parts of a multi-part message are separated by the string in the MultipartBoundary property. This string must be chosen so that it does not occur in the body of any message part. The underscore character ('_') does not appear in Base64 encoded data so if you use that character somewhere in your boundary you do not need to be concerned about what will be in the body of a Base64 encoded message part.

Data Type

String

NewMessage Method

Description

Creates a new, empty message.

Syntax

*object.***NewMessage**

The syntax of the **NewMessage** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

Discards the contents of the current message and all of its parts, if any, and creates a new empty e-mail message.

Part Property

See Also Error Codes Frequently Asked Questions

Description

Selected part of a multi-part message.

Syntax

*object.***Part** [= *long*]

The syntax of the **Part** property has these parts:

Part	Description
object	A Mail control.
long	A long integer that specifies an individual part of a multi-part message.

Remarks

This property is used to select one part of a multi-part message. Typically you will set this property before using the CreatePart method, or before using the Descend method to descend into one part of a multi-part message.

Data Type

Integer (long)

Parts Property

See Also Error Codes Frequently Asked Questions

Description

Number of parts in the current multi-part message.

Syntax

*object.***Parts**

The syntax of the **Parts** property has these parts:

Part	Description
object	A Mail control.

Remarks

A MIME message may contain multiple parts. The Parts property tells you how many parts are contained in the current message.

This property is read-only and only available at run-time.

Data Type

Integer (long)

PopMessageCount Property

See Also Error Codes Frequently Asked Questions

Description

Number of new messages available from the POP3 server.

Syntax

*object.***PopMessageCount**

The syntax of the **PopMessageCount** property has these parts:

Part	Description
object	A Mail control.

Remarks

After the Mail control connects to a POP3 server, this property tells you how many new messages are contained in the user's POP3 mailbox.

This property is read-only and only available at run-time.

Data Type

Integer

PopMessageSizes Property

See Also Error Codes Frequently Asked Questions

Description

Number of bytes in each POP3 message.

Syntax

*object.***PopMessageSizes** [= *index*]

The syntax of the **PopMessageSizes** property has these parts:

Part	Description
object	A Mail control.
index	An integer that identifies a message.

Remarks

After you have connected to a POP3 server, the PopMessageCount property tells you how many e-mail messages are in the user's POP3 mailbox. The PopMessageSizes array will then have one element for each message. Each element tells you how many bytes are in each e-mail message. When downloading messages you can use this information together with the information provided in the Progress event to display a progress bar.

This property is read-only and only available at run-time.

Data Type

Integer (long)

PopPort Property

See Also Error Codes Frequently Asked Questions

Description

Specifies POP server's port on the Host.

Syntax

*object.***PopPort** [= *port*]

The syntax of the **PopPort** property has these parts:

Part	Description
object	A Mail control.
port	A long integer that specifies the POP server's port number.

Remarks

This property determines the port number to use when accessing your POP server. Normally you can use the default setting.

Sometimes security measures (firewalls, wormholes, etc.) require you to use a different port. This property allows you to change the port number that the Mail control uses to accommodate the security.

PopPort is used when the ConnectType property is MailConnectTypePOP3 (1).

Data Type

Integer (long)

Progress Event

See Also Error Codes Frequently Asked Questions

Description

Fired as lengthy operations progress.

Syntax

Sub *object*_**Progress(**[*index* **As Integer**,] *bytecount* **As Long**, *bytestotal* **As Long)**

The syntax of the **Progress** event has these parts:

Part	Description
object	A Mail control.
index	An integer that identifies a control if it's in a control array.
bytecount	A long integer that specifies the number of bytes sent or received.
bytestotal	A long integer that specifies the total number of bytes to send or receive.

Remarks

The Progress event is fired while e-mail is being sent or received. This event lets you know how close to completion the operation is and may be used to display a progress bar or other status indicator.

Read Method

See Also Error Codes Frequently Asked Questions

Description

Used to do "raw" reads of server responses.

Syntax

*object.***Read**

The syntax of the **Read** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

This control has a pair of methods, Read and Write, and corresponding properties, ReadData and WriteData, which can be used to communicate directly with the server. These methods and properties are provided so that you can handle non-standard protocols which may be implemented on special-purpose servers.

Used to read the server's response after a command has been sent using the Write method. The Done event is fired when data arrives from the server after the Write method has been invoked if the control is in non-blocking mode (see Blocking).

ReadData Property

See Also Error Codes Frequently Asked Questions

Description

Data received from the Read method or its Action equivalent.

Syntax

*object.***ReadData**

The syntax of the **ReadData** property has these parts:

Part	Description
object	A Mail control.

Remarks

This control has a pair of methods, Read and Write, and corresponding properties, ReadData and WriteData, which can be used to communicate directly with the server. These methods and properties are provided so that you can handle non-standard protocols which may be implemented on special-purpose servers.

There may be circumstances where you need to interact directly with the server. This can happen if a particular server has non-standard extensions you choose to use. The Read method lets you directly retrieve data sent by the server. Upon successful completion of the Read method (or the Action property equivalent) the ReadData property will contain whatever the server sent.

Data Type

String

ReadMessage Method

See Also Error Codes Frequently Asked Questions

Description

Reads a message and makes it available through the control's properties.

Syntax

*object.***ReadMessage** *flags*, *filenameOrID*

The syntax of the **ReadMessage** method has these parts:

Part	Description
object	Required. A Mail control.
flags	Optional. An integer that specifies the source of the message.
filenameOrID	Optional. A string expression that contains either a filename or a message ID (depending on the *flags* parameter).

Remarks

The Flags property (or the *flags* parameter) specifies the source of the e-mail message to be read. You can read messages from files, buffers, or from a POP3 host.

Flag	Description
MailSrcIsBuffer	Buffer contains message to read
MailSrcIsFilename	SrcFilename has the name of a file containing the message to read
MailSrcIsHost	Host server is used to retrieve message specified by MessageID
MailReadHeaderOnly	Only the header lines are read from a message. When this flag is set, only the headers are retrieved. Everything normally associated with an e-mail message will be as it is without the MailReadHeadersOnly flag, except that the message will contain no body text of any kind.

Important Note for Mail

For most commands which interact with either a POP or SMTP server, the Timeout property specifies the total time to wait for the command to complete. Note that timeout values are specified in seconds, not milliseconds.

With either the ReadMessage or WriteMessage commands, the Timeout property specifies an inactivity time. If data does not arrive (or cannot be sent) within the time specified by the Timeout property, the command is aborted.

If you set some reasonable value for Timeout, 30 seconds for instance, it will most likely work for connecting, sending and retrieving messages, and all other mail commands.

SmtpPort Property

See Also Error Codes Frequently Asked Questions

Description

Specifies SMTP server's port on the Host.

Syntax

*object.***SmtpPort** [= *port*]

The syntax of the **SmtpPort** property has these parts:

Part	Description
object	A Mail control.
port	A long integer that specifies the SMTP server's port number.

Remarks

This property determines the port number to use when accessing your SMTP server. Normally you can use the default setting.

Sometimes security measures (firewalls, wormholes, etc.) require you to use a different port. This property allows you to change the port number that the Mail control uses to accommodate the security.

SmtpPort is used when the ConnectType property is MailConnectTypeSMTP (0).

Data Type

Integer (long)

SrcFilename Property

Description

Source filename for some methods.

Syntax

*object.***SrcFilename** [= *string*]

The syntax of the **SrcFilename** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies a filename.

Remarks

A number of operations, such as Encode and Decode, may use a source file. When that is the case the source filename is given by the SrcFilename property.

Data Type

String

Subject Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

Subject of the e-mail message.

Syntax

*object.***Subject** [= *string*]

The syntax of the **Subject** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that specifies the subject of a message.

Remarks

E-mail messages typically have a "Subject:" header line. When e-mail is received, the "Subject:" header line is parsed and the subject itself is stored in the Subject property. Before sending e-mail, you may set the Subject property.

Data Type

String

Timeout Property

See Also Error Codes Frequently Asked Questions

Description

Time in seconds to wait for an operation to complete.

Syntax

*object.***Timeout**

The syntax of the **Timeout** property has these parts:

Part	Description
object	A Mail control.

Remarks

Methods which require interaction with a server may hang indefinitely due to a slow or non-responsive server. You can set the maximum time to wait for any operation to complete by assigning a non-zero number of seconds to the Timeout property. When the Timeout property is set to zero there will be no time-out.

Data Type

Integer

To Property

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

Intended message recipient(s).

Syntax

*object.***To** [= *string*]

The syntax of the **To** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that contains the receipient(s) of the current message.

Remarks

Three properties are used to specify the recipients of any given e-mail message, the To, CC, and BCC properties. All recipients will see the recipients in the To and CC properties in the header of the copy of the e-mail message which they receive. Recipients listed in the BCC field will not be known to other recipients.

Multiple recipients may be specified and, if so, they must be separated by commas. The following are examples of valid addresses:

```
zane@mabry.com
<zane@mabry.com>
Zane Thomas <zane@mabry.com>
```

If you choose to include a full name as well as an e-mail address, as is done in the third example, you must enclose the actual e-mail address in angle brackets.

You should not use other formats for specifying an e-mail address. The Mail control will attempt to extract the actual e-mail address from the string specified. You may find, however, that the Mail control does not recognize them.

Data Type

String

Version Property

Error Codes Frequently Asked Questions

Description

Returns the version of the control.

Syntax

*object.***Version**

The syntax of the **Version** property has these parts:

Part	Description
object	A Mail control.

Remarks

This property holds the current version of the control. It is read-only and available at both design-time and run-time.

Data Type

String

Write Method

See Also Error Codes Frequently Asked Questions

Description

Sends a string to the server.

Syntax

*object.***Write**

The syntax of the **Write** method has these parts:

Part	Description
object	Required. A Mail control.

Remarks

This control has a pair of methods, Read and Write, and corresponding properties, ReadData and WriteData, which can be used to communicate directly with the server. These methods and properties are provided so that you can handle non-standard protocols which may be implemented on special-purpose servers.

Used to write a string to the server. This method is useful for sending non-standard commands or for invoking some of the more obscure, or even non-standard, commands. When the server's response comes back, the Done event is fired if the control is in non-blocking mode.

All strings must be terminated with a CR/LF (Chr(13) & Chr(10)) as part of the string.

WriteData Property

See Also Error Codes Frequently Asked Questions

Description

Data to send to server

Syntax

*object.***WriteData** [= *string*]

The syntax of the **WriteData** property has these parts:

Part	Description
object	A Mail control.
string	A string expression that holds data to send to the FTP server.

Remarks

This control has a pair of methods, Read and Write, and corresponding properties, ReadData and WriteData, which can be used to communicate directly with the server. These methods and properties are provided so that you can handle non-standard protocols which may be implemented on special-purpose servers.

This control handles all aspects of the usual protocols. However, there may be circumstances where you need to interact directly with a server. When this is the case, you can send commands directly to the server by assigning a command string to the WriteData property and using the Write method (or its Action property equivalent).

All strings must be terminated with a CR/LF (Chr(13) & Chr(10)) as part of the string.

Data Type

String

WriteMessage Method

See Also | Error Codes | Simple Message Send Example | Frequently Asked Questions

Description

Writes an e-mail message either to a file, buffer, or an SMTP server.

Syntax

*object.***WriteMessage** *flags*, *filename*

The syntax of the **WriteMessage** method has these parts:

Part	Description
object	Required. A Mail control.
flags	Optional. An integer that specifies the destination of the message.
filename	Optional. A string expression that holds the filename (only used if *flags* contains MailDstIsFilename).

Remarks

Writes a fully formatted MIME message to the destination specified by the Flags property (or by the optional *flags* parameter when using the OCX version).

The Flags property (or optional *flags* parameter when using the OCX version) determines the destination of the message:

Flag	Meaning
MailDstIsBuffer	Formatted message is written to the Buffer property
MailDstIsFilename	Formatted message is written to the file named by the DstFilename property
MailDstIsHost	Formatted message is written to the mail host .

Important Note for Mail

For most commands which interact with either a POP or SMTP server, the Timeout property specifies the total time to wait for the command to complete. Note that timeout values are specified in seconds, not milliseconds.

With either the ReadMessage or WriteMessage commands, the Timeout property specifies an inactivity time. If data does not arrive (or cannot be sent) within the time specified by the Timeout property, the command is aborted.

If you set some reasonable value for Timeout, 30 seconds for instance, it will most likely work for connecting, sending and retrieving messages, and all other mail commands.

Getting Custom Controls Written

If you or your organization would like to have custom controls written, you can contact us at the following:

Mabry Software, Inc.
Post Office Box 31926
Seattle, WA 98103-1926

Phone: 206-634-1443
Fax: 206-632-0272 or 206-364-3196

Internet: mabry@mabry.com

You can also contact Zane Thomas. He can be reached at:

Zane Thomas
Post Office Box 121
Indianola, WA 98342

Internet: zane@mabry.com

Licensing Information

Legalese Version

Mabry Software grants a license to use the enclosed software to the original purchaser. Copies may be made for back-up purposes only. Copies made for any other purpose are expressly prohibited, and adherence to this requirement is the sole responsibility of the purchaser.

Customer written executable applications containing embedded Mabry products may be freely distributed, without royalty payments to Mabry Software, provided that such distributed Mabry product is bound into these applications in such a way so as to prohibit separate use in design mode, and that such Mabry product is distributed only in conjunction with the customers own software product. The Mabry Software product may not be distributed by itself in any form.

Neither source code for Mabry Software products nor modified source code for Mabry Software products may be distributed under any circumstances, nor may you distribute .OBJ, .LIB, etc. files that contain our routines. This control may be used as a constituent control only if the compound control thus created is distributed with and as an integral part of an application. Permission to use this control as a constituent control does not grant a right to distribute the license (LIC) file or any other file other than the control executable itself. This license may be transferred to a third party only if all existing copies of the software and its documentation are also transferred.

This product is licensed for use by only one developer at a time. Mabry Software expressly prohibits installing this product on more than one computer if there is any chance that both copies will be used simultaneously. This restriction also extends to installation on a network server, if more than one workstation will be accessing the product. All developers working on a project which includes a Mabry Software product, even though not working directly with the Mabry product, are required to purchase a license for that Mabry product.

This software is provided as is. Mabry Software makes no warranty, expressed or implied, with regard to the software. All implied warranties, including the warranties of merchantability and fitness for a particular use, are hereby excluded.

MABRY SOFTWARE'S LIABILITY IS LIMITED TO THE PURCHASE PRICE. Under no circumstances shall Mabry Software or the authors of this product be liable for any incidental or consequential damages, nor for any damages in excess of the original purchase price.

To be eligible for free technical support by telephone, the Internet, CompuServe, etc. and to ensure that you are notified of any future updates, please complete the enclosed registration card and return it to Mabry Software.

English Version

We require that you purchase one copy of a control per developer on a project. If this is met, you may distribute the control with your application royalty free. You may never distribute the LIC file. You may not change the product in any way that removes or changes the requirement of a license file.

We encourage the use of our controls as constituent controls when the compound controls you create are an integral part of your application. But we don't allow distribution of our controls as constituents of other controls when the compound control is not part of an application. The reason we need to have this restriction is that without it someone might decide to use our control as a constituent, add some trivial (or even non-trivial) enhancements and then sell the compound control. Obviously there would be little difference between that and just plain reselling our control.

If you have purchased the source code, you may not re-distribute the source code either (nor may you copy it into your own project). Mabry Software retains the copyright to the

source code.

Your license is transferable. The original purchaser of the product must make the transfer request. Contact us for further information.

The sample versions of our products are intended for evaluation purposes only. You may not use the sample version to develop completed applications.