Help for FTP

Properties Events Methods Frequently Asked Questions

Registration Information

Order Form

Getting Custom Controls Written

Licensing Information

Description

The FTP VBX/OCX custom control provides the complete File Transfer Protocol. It allows you to send and receive files from FTP servers, get directory information from a server, and manage directories on a server. You can use the Mabry FTP control to automatically transfer files to/from a host when a user enters or needs data. FTP requires a TCP/IP connection. It complies with RFC 959 as well as RFC 1579 (the firewall-friendly FTP specification).

File Name

MFTP1.VBX, MFTP32.OCX

ActiveX / OCX Object Name

Mabry.FTPCtrl

ActiveX Compatility

VB 4.0 (32-bit) and 5.0, Access 97

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

mFTP

VBX Compatibility

VB 2.0, 3.0 and 4.0 (16-bit)

VBX Built With

Microsoft Visual C++ v1.5

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

FTP Properties

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

*__Account__ Property
*__Action__ Property
*__AllocBytes__ Property
*__Blocking__ Property
*__Debug__ Property
*__Directory__ Property
*__DisablePasv__ Property
*__DstFilename__ Property
*__Host__ Property
*__LastError__ Property
*__LogonName__ Property
*__LogonPassword__ Property
*__Pattern__ Property
*__Port__ Property
*__QuoteCmd__ Property
*__ReadData__ Property
*__SrcFilename__ Property
*__State__ Property
*__Timeout__ Property
*__Type__ Property
*__Version__ Property
*__WriteData__ Property

Close

FTP Events

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

*__AsyncError__ Event
*__Connected__ Event
*__Debug__ Event
*__DirItem__ Event
*__Done__ Event
*__Progress__ Event

Close

FTP Methods

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

*<u>**Abort** Method</u>
*<u>**AboutBox** Method</u>
*<u>**Allocate** Method</u>
*<u>**Append** Method</u>
*<u>**ChangeDir** Method</u>
*<u>**Connect** Method</u>
*<u>**CreateDir** Method</u>
*<u>**Delete** Method</u>
*<u>**DeleteDir** Method</u>
*<u>**Disconnect** Method</u>
*<u>**GetCurrentDir** Method</u>
*<u>**GetDirList** Method</u>
*<u>**GetFile** Method</u>
*<u>**GetFilenameList** Method</u>
*<u>**ParentDir** Method</u>
*<u>**PutFile** Method</u>
*<u>**PutUniqueFile** Method</u>
*<u>**Quote** Method</u>
*<u>**Read** Method</u>
*<u>**Reinitialize** Method</u>
*<u>**Rename** Method</u>
*<u>**SendType** Method</u>
*<u>**Write** Method</u>

Close

Frequently Asked Questions

How do I use the CHMOD command with the Mabry FTP control?

Frequently Asked Questions

Make sure you have the latest version of the control (v5.00.006 or higher). These support the Quote command.

So, you'd simply set the QuoteCmd property and issue the Quote method.   For example:

```
mFTP1.QuoteCmd = ""CHMOD 777 " & filename
mFTP1.Quote
```

How can I be sure that a file transfer succeeded in synchronous mode (Blocking = True)?

You can assume a successful file transfer if the LastError property is zero (0) when the transfer process completes. If the LastError property is not zero, an error occurred. See the LastError property in the FTP help file or your Internet Pack manual for a list of errors and their descriptions.

How do I connect to a server?

Frequently Asked Questions

To connect to a server you need to set the Host, LogonName, LogonPassword properties. The Host property needs to hold the name of the FTP server you want to connect to, LogonName holds your user name, and LogonPassword holds your password.

For an anonymous FTP connection, you set LogonName to "anonymous" and LogonPassword to your e-mail address.

Once these properties are set, set the Action property to FtpActionConnect (or, with the OCX version, use the Connect method).

```
Ftp1.Blocking = True
Ftp1.Host = "ftp.mabry.com"
Ftp1.LogonName = "anonymous"
Ftp1.LogonPassword = "you@your.domain.com"
Ftp1.Action = FtpActionConnect
```

How do I change directories on the server?

Frequently Asked Questions

First, you must be connected to a server. Then, set the Directory property to the path of the desired directory. Last, set the Action property to FtpActionChangeDir (or, with the OCX version, use the ChangeDir method).

```
    Ftp1.Directory = "somedir"
    Ftp1.Action = FtpActionChangeDir
```

How do I get a file from the server?

```
Ftp1.SrcFilename = "hostfilename"
Ftp1.DstFilename = "c:\local.cpy"
Ftp1.Action = FtpActionGetFile
```

How do I put a file onto the server?

Frequently Asked Questions

```
Ftp1.SrcFilename = "c:\local.cpy"
Ftp1.DstFilename = "hostfilename"
Ftp1.Action = FtpActionPutFile
```

When the password, or username are incorrect, the internet control returns an error. What happens next? Does the connection stay open? Does it disconnect?

Frequently Asked Questions

The short answer to your question is that when a server rejects either a username or password, the connection is not completely established yet.

What action do you recommend to be taken when this error crops up, should a disconnection be done, or an abort?

Frequently Asked Questions

When you've trapped either error, you can simply re-issue the Connect method again or you may wish to prompt the user to change the incorrect setting. You would not need to issue the Abort or Disconnect method since that connection wasn't established and that'll just throw a run-time error that you'll need to handle also.

The Abort and Disconnect methods are really used once the client has been authenticated by the server and they've entered the "Transaction" state (as defined in the RFC).

How do I get a file's size using the Mabry FTP control?

Frequently Asked Questions

There are a couple of ways of getting the file size.

1. The Hard Way. Use the GetDirList to retrieve all of the info in the directory and parse the returned DirItem.

2. The Easy Way. The Check to see if the FTP server supports a SIZE command. You can use the QuoteCmd property and Quote command to send such commands directly to the server.

The server should return a success/failure code and then the file size, ie - 213 15825, where 213 is the server result code and 15825 is the size in bytes).

For example, to get the file size with Blocking = True:

```
mFTP1.QuoteCmd = "SIZE test.txt"
mFTP1.Quote
If Left(mFTP1.ReadData,1) > "2" Then
    ' Error Handling Code
Else
    FileSize = Val(Trim(Mid(mFTP1.ReadData,4)))
End If
```

To get the file size with Blocking = False, there are two places you need to put code. You need to issue the command in one place and you need to retrieve the response in the Done event.

```
mFTP1.QuoteCmd = "SIZE test.txt"
mFTP1.Quote
```

Here's the code for the Done event:

```
Sub mFTP1_Done
    If Left(mFTP1.ReadData,1) > "2" Then
        ' Error Handling Code
    Else
        FileSize = Val(Trim(Mid(mFTP1.ReadData,4)))
    End If
End Sub
```

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!

How can I use the FTP control to connect to an FTP server that is behind a firewall or proxy server?

Frequently Asked Questions

It varies depending upon the specific configuration of the firewall or proxy server. In some cases, a combination of the Account (for specifying certain user or file access privileges) and Port (for specifying non-standard port values) properties is sufficient.

The FTP protocol requires that data transfers are preceeded by either a PORT (active) or PASV (passive) command.   The PASV command must be used with servers that are behind firewalls.   However, some server/firewall combinations do not respond to the PASV command as required by the applicable RFCs. In those cases, setting the DisablePasv property to True prior to establishing the connection will force the control to use the PORT command and establish an active connection.

Finally, the Mabry FTP control conforms to the FTP document RFC 959 and is designed to handle all FTP commands. Should the above methods fail, the Quote command can be used for sending special commands directly to the FTP server (once connected) and the Mabry's Asocket control could possibly be used to handle connections for commands outside of FTP, such as establishing a connection through a proxy.

When I connect to my FTP server it hangs and ultimately I get a WinSock error, usually 10004. Why?

Frequently Asked Questions

By default, the control attempts to establish a passive (PASV) connection and will switch to an active (PORT) connection if the server rejects the passive command. Some servers respond positively to the PASV command, but then do not actually listen on the specified port. Make sure you are using version 5.00.010 or later of the Mabry FTP control, set the DisablePasv property to True and attempt the connection again.

What determines when the Progress event fires?   It seems to fire less frequently on transfer than on receives.   Sometimes it doesn't fire at all for short files.

For very short files the progress event may not fire.   This is normal.

Frequency of send/receive progress events depends entirely upon the stack and the net. It's no surprise that receive events are more frequent than send events since the stack accepts large data buffers for transmission, but they may be broken up into smaller chunks when sent out on the wire.

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?

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?

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?

Frequently Asked Questions

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?

Frequently Asked Questions

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?

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

FTP 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 FTP (control only) is US$35 (US$40 for International orders).   The cost of FTP and the C/C++ source code (of the control itself) is US$90 (US$95 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.   FTP's SWREG ID number is 6393.   The source code version's ID number is 9060.   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.

FTP 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 FTP?

____

Name: ____

Ship to: ____

____

____

____

Phone: ____

Fax: ____

E-Mail: ____

MC/VISA/AMEX: ______________________________ exp. _____________

P.O. # (if any): _______________ Signature ___________________________

qty ordered	____	REGISTRATION $35.00 ($40.00 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 $90.00 ($95.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $5.00 per order for shipping and handling.

ChangeDir Example

This snippet of code shows how to change directories, once connected to a server.   You can change the setting of the Directory propery to change to the directory you need.

```
    Ftp1.Directory = "directoryname"
    Ftp1.Action = FtpActionChangeDir
```

Note: We used the Action property in this example because it is compatible for both the VBX and OCX-32 versions of the Mabry FTP control.   If you want to use the ChangeDir method (only available in the OCX-32 version of the control), you can use the following code:

```
    Ftp1.Directory = "directoryname"
    Ftp1.ChangeDir
```

Close Copy Print

Connect Example

This snippet of code shows how to connect to Mabry's FTP server.   You can change the settings of the Host, LogonName and LogonPassword properties to connect to other servers.

```
    Ftp1.Blocking = True
    Ftp1.Host = "ftp.mabry.com"
    Ftp1.LogonName = "anonymous"
    Ftp1.LogonPassword = "you@yourdomain.com"
    Ftp1.Action = FtpActionConnect
```

Note: We used the Action property in this example because it is compatible for both the VBX and OCX-32 versions of the Mabry FTP control.   If you want to use the Connect method (only available in the OCX-32 version of the control), you can use the following code:

```
    Ftp1.Blocking = True
    Ftp1.Host = "ftp.mabry.com"
    Ftp1.LogonName = "anonymous"
    Ftp1.LogonPassword = "you@yourdomain.com"
    Ftp1.Connect
```

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.
	20200	Command okay.
	20202	Command not implemented, superfluous at this site.
	20211	System status, or system help reply.
	20212	Directory status.
20213	File status.	
	20214	Help message. On how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.
	20215	NAME system type. Where NAME is an official system name from the list in the Assigned Numbers document.
	20220	Service ready for new user.
	20221	Service closing control connection. Logged out if appropriate.
	20225	Data connection open; no transfer in progress.

20226	Closing data connection. Requested file action successful (for example, file transfer or file abort).
20227	Entering Passive Mode (h1,h2,h3,h4,p1,p2).
20230	User logged in, proceed.
20250	Requested file action okay, completed.
20257	"PATHNAME" created.
20421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
20425	Can't open data connection.
20426	Connection closed; transfer aborted.
20450	Requested file action not taken. File unavailable (e.g., file busy).
20451	Requested action aborted: local error in processing.
20452	Requested action not taken. Insufficient storage space in system.
20500-20599	Errors reported by the server.
20501	Syntax error in parameters or arguments.
20502	Command not implemented.
20503	Bad sequence of commands.
20504	Command not implemented for that parameter.
20530	Not logged in.
20532	Need account for storing files.
20550	Requested action not taken. File unavailable (e.g., file not found, no access).
20551	Requested action aborted: page type unknown.
20552	Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
20553	Requested action not taken. File name not allowed.
20601	Unexpected server response.
20600	Internal control state error.
20602	Already connected to server.
20603	Busy executing asynchronous command.
20604	Can't change blocking mode, busy or connected to server.
20605	Operation timed out.
20606	Invalid user name.

20607	Invalid password.
20608	Could not open file.

See Also

Blocking Property

See Also

Action Property
LogonName Property
LogonPassword Property
Connect Method

See Also

LastError Property

AsyncError Event

Done Event

See Also

Action Property

AllocBytes Property

LastError Property

See Also

Allocate Method

PutFile Method

See Also

Action Property

DstFilename Property

SrcFilename Property

Done Event

Allocate Method

PutFile Method

See Also

Action Property

Blocking Property

Done Event

See Also

Action Property

AsyncError Event

Done Event

Connect Method

Disconnect Method

See Also

Action Property

Directory Property

Done Event

GetDirList Method

ParentDir Method

See Also

Account Property

Action Property

Host Property

LogonName Property

LogonPassword Property

Connected Event

Done Event

Disconnect Method

Reinitialize Method

See Also

Action Property

State Property

Connect Method

Disconnect Method

Reinitialize Method

See Also

Action Property

Directory Property

Done Event

ChangeDir Method

DeleteDir Method

ParentDir Method

See Also

Debug Event

See Also

Debug Property

See Also

SrcFilename Property

Done Event

DeleteDir Method

See Also

Action Property

Directory Property

Done Event

ChangeDir Method

CreateDir Method

See Also

Action Property

ChangeDir Method

GetCurrentDir Method

See Also

Action Property

GetDirList Method

GetFilenameList Method

See Also

Connected Event
Abort Method
Connect Method
Reinitialize Method

See Also

Blocking Property

AsyncError Event

See Also

SrcFilename Property

GetFile Method

PutFile Method

Rename Method

See Also

Action Property

Directory Property

ChangeDir Method

CreateDir Method

DeleteDir Method

ParentDir Method

See Also

Action Property

DirItem Event

Done Event

GetFilenameList Method

See Also

Action Property

DstFilename Property

SrcFilename Property

Done Event

Progress Event

Allocate Method

Append Method

PutFile Method

SendType Method

See Also

Action Property

Pattern Property

DirItem Event

Done Event

GetDirList Method

See Also

Port Property

Connect Method

See Also

Action Property

AsyncError Event

Done Event

See Also

Account Property
Action Property
LogonPassword Property
Connect Method

See Also

Account Property

Action Property

LogonName Property

Connect Method

See Also

Action Property

Done Event

ChangeDir Method

CreateDir Method

GetDirList Method

See Also

Action Property

GetDirList Method

GetFilenameList Method

See Also

Host Property

Connect Method

See Also

Action Property

GetFile Method

PutFile Method

See Also

Action Property
DstFilename Property
SrcFilename Property
Done Event
Progress Event
Allocate Method
Append Method
GetFile Method
SendType Method

See Also

Action Property

SrcFilename Property

Done Event

Progress Event

Allocate Method

GetFile Method

PutFile Method

SendType Method

See Also

QuoteCmd Property
ReadData Property
WriteData Property
Done Event
Read Method
Write Method

See Also

ReadData Property
WriteData Property
Done Event
Quote Method
Read Method
Write Method

See Also

ReadData Property

WriteData Property

Done Event

Write Method

See Also

Action Property

WriteData Property

Done Event

Read Method

See Also

Account Property
Action Property
Host Property
LogonName Property
LogonPassword Property
Connected Event
Done Event
Connect Method
Disconnect Method

See Also

Action Property
DstFilename Property
SrcFilename Property
Done Event
Append Method
PutFile Method
SendType Method

See Also

Action Property

Type Property

Done Event

GetFile Method

PutFile Method

See Also

Action Property

GetFile Method

PutFile Method

Rename Method

See Also

Action Property

Connect Method

Disconnect Method

Reinitialize Method

See Also

AsyncError Event

Done Event

See Also

Action Property

GetFile Method

PutFile Method

See Also

ReadData Property

WriteData Property

Done Event

Read Method

See Also

Action Property

ReadData Property

Write Method

Abort Method

See Also Error Codes Frequently Asked Questions

Description

Cancels commands in progress. Only valid when Blocking is False.

Syntax

*object.***Abort**

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

Part	**Description**
object	Required. An FTP control.

Remarks

Used to cancel commands when the Blocking property is False. File transfer commands may be canceled using the Abort method regardless of the state of the Blocking property. However, it is recommended, if your user-interface requires that the user be able to cancel transfers, that the Blocking property be set to False so that user actions can be rapidly serviced.

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. An FTP control.

Remarks

This method displays the About Box for this control which includes copyright information.

Account Property

See Also   Error Codes   Frequently Asked Questions

Description

Account name to use when connecting to the FTP server.

Syntax

*object.***Account** [= *string*]

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

Part	**Description**
object	An FTP control.
string	A string expression that holds an account name for logging on to the FTP server.

Remarks

Some servers may require that an account name be supplied in addition to the LogonName and LogonPassword.   For servers which require an account, this property must be set before invoking the Connect method.

Data Type

String

Action Property

See Also   Error Codes   Frequently Asked Questions

Description

Causes control to initiate a command / method.

Syntax

*object.***Action** [= *action*]

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

Part	Description
object	An FTP control.
action	An integer that specifies which method to execute.

Remarks

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

Constant	Value	Description
FtpActionNone	0	No action
FtpActionAbort	1	Abort current action (Abort)
FtpActionAllocate	2	Allocate server storage (Allocate)
FtpActionAppend	3	Append to server file (Append)
FtpActionChangeDir	4	Change server directory (ChangeDir)
FtpActionConnect	5	Connect to server (Connect)
FtpActionCreateDir	6	Create server directory (CreateDir)
FtpActionDelete	7	Delete server file (Delete)
FtpActionDeleteDir	8	Delete server directory (DeleteDir)
FtpActionDisconnect	9	Disconnect from server (Disconnect)
FtpActionGetCurrentDir	10	Get current server directory (GetDirList)
FtpActionGetDirList	11	Get server directory listing (GetDirList)
FtpActionGetFile	12	Get server file (GetFile)
FtpActionGetFilenameList	13	Get server filename list (GetFilenameList)
FtpActionParentDir	14	Change to server parent directory (ParentDir)
FtpActionPutFile	15	Send file to server (PutFile)
FtpActionPutUniqueFile	16	Send file to server, server chooses name (PutUniqueFile)
FtpActionSendType	17	Set transfer type (SendType)
FtpActionRead	18	Read response from server (Read)
FtpActionRename	19	Rename server file (Rename)
FtpActionReinitialize	20	Reinitialize session (Reinitialize)
FtpActionWrite	21	Write command string to server (Write)
FtpActionQuote	22	Quote command (Quote)

Data Type

Integer (enumerated)

Allocate Method

See Also   Error Codes   Frequently Asked Questions

Description

Allocates file space on the FTP server for storage.

Syntax

*object.***Allocate** *bytecount*

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

Part	**Description**
object	Required. An FTP control.
bytecount	Optional. A long integer expression holding the number of bytes to allocate. If not specified, the AllocBytes property is used.

Remarks

Some FTP servers require you to allocate storage for a file before sending it.   The Allocate method is used to perform this allocation, requesting the server to allocate the number of bytes of storage specified by the AllocBytes property.

Many servers do not actually require that this command be used.   The FTP specification requires servers to ignore this command if they do not require/support it.

To be safe under all circumstances, you should always use Allocate before sending any file and then ignore any 205xx errors (see the LastError property) which may be returned.   Do not, however, ignore 204xx errors since the server may be indicating that it cannot fulfill the request at this time.

AllocBytes Property

See Also Error Codes Frequently Asked Questions

Description

Number of bytes to allocate on the server's storage system.

Syntax

*object.***AllocBytes** [= *long*]

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

Part	**Description**
object	An FTP control.
long	A long integer that specifies the number of bytes to pre-allocate on the FTP server.

Remarks

Some servers require that you allocate storage for a file before sending it with the PutFile method. Set the AllocBytes property to the source file size before invoking the Allocate method.

Data Type

Integer (long)

Append Method

See Also   Error Codes   Frequently Asked Questions

Description

Appends to a file on the FTP server.

Syntax

*object.***Append** *srcfilename*, *dstfilename*

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

Part	**Description**
object	Required. An FTP control.
srcfilename	Optional. A string expression holding the source filename. If not specified, the SrcFilename property is used.
dstfilename	Optional. A string expression specifying the destination filename. If not specified, the DstFilename property is used.

Remarks

Appends data from the local file named by the *srcfilename* parameter to the FTP server's file, named by the *dstfilename* parameter. If *dstfilename* does not exist, it is created on the server.

Some servers may require you to use the Allocate method prior to starting a file transfer.

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	An FTP 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.

Blocking Property

See Also Error Codes Frequently Asked Questions

Description

Determines if methods and actions are blocking.

Syntax

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

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

Part	**Description**
object	An FTP 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.

The Blocking property must be set prior to using the Connect method and cannot be subsequently used until the Disconnect method has completed.

Data Type

Integer (boolean)

ChangeDir Method

See Also Error Codes Frequently Asked Questions

Description

Changes the current directory on the FTP server.

Syntax

*object.***ChangeDir** *directory*

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

Part	**Description**
object	Required. An FTP control.
directory	Optional. A string expression holding the desired new directory. If not specified, the Directory property is used.

Remarks

Changes the server's current directory to that specified by the *directory* parameter.

Connect Method

See Also   Error Codes   Frequently Asked Questions

Description

Connects to an FTP server.

Syntax

object.**Connect** *username*, *password*, *account*

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

Part	**Description**
object	Required. An FTP control.
username	Optional. A string expression holding the user's name. If not specified, the LogonName property is used.
password	Optional. A string expression specifying the user's password. If not specified, the LogonPassword property is used.
account	Optional. A string expression specifying the user's account name. If not specified, the Account property is used.

Remarks

Connects to FTP server named by the Host property. The *username* and *password* parameters are used during connection. The these parameters are not specified, the LogonName and LogonPassword properties are used.

Some hosts may also require a valid *account* parameter.

Upon connection, the FTP control fires the Connected event.

Connected Event

See Also Error Codes Frequently Asked Questions

Description

Fires when connection state changes.

Syntax

Sub *object*_**Connected(**[*index* **As Integer**,] *connectstate* **As Boolean)**

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

Part	**Description**
object	An FTP control.
index	An integer that identifies a control if it's in a control array.
connectstate	A boolean flag that specifies whether or not the control is connected to an FTP server.

Remarks

Fired when the connection state changes.   The *connectstate* parameter is True after a successful Connect or Reinitialize method execution.

The *connectstate* parameter is False after the Disconnect method completes or if the server closes the connection.

CreateDir Method

See Also Error Codes Frequently Asked Questions

Description

Creates a directory on the FTP server.

Syntax

*object.***CreateDir** *directory*

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

Part	**Description**
object	Required. An FTP control.
directory	Optional. A string expression holding the desired new directory. If not specified, the Directory property is used.

Remarks

Creates a remote directory named by the *directory* parameter.

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	An FTP 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	An FTP 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.

Delete Method

See Also Error Codes Frequently Asked Questions

Description

Delete's a file from the FTP server.

Syntax

*object.***Delete** *dstfilename*

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

Part	**Description**
object	Required. An FTP control.
dstfilename	Optional. A string expression that specifies a filename to delete from the FTP server.

Remarks

This method deletes a file, specified by *dstfilename*, from the FTP server.

DeleteDir Method

See Also Error Codes Frequently Asked Questions

Description

Deletes a directory on the FTP server.

Syntax

*object.***DeleteDir** *directory*

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

Part	**Description**
object	Required. An FTP control.
directory	Optional. A string expression holding the desired new directory. If not specified, the Directory property is used.

Remarks

Deletes a directory named by the *directory* parameter.

Directory Property

See Also   Error Codes   Frequently Asked Questions

Description

Current directory on FTP server.

Syntax

*object.***Directory** [= *directory*]

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

Part	**Description**
object	An FTP control.
directory	A string expression that specifies a directory on the FTP server.

Remarks

You set this property to the FTP server directory you wish to change to before using the ChangeDir method or Action.   This property is set to the server's current directory upon completion of the GetCurrentDir method or Action.

Data Type

String

DirItem Event

See Also   Error Codes   Frequently Asked Questions

Description

Fired for each directory line received.

Syntax

Sub *object*_**DirItem(**[*index* **As Integer**,] *item* **As String)**

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

Part	**Description**
object	An FTP control.
index	An integer that identifies a control if it's in a control array.
item	A string expression that contains a directory list item retrieved from the FTP server.

Remarks

Fired for each line returned from the server for either the GetDirList or GetFilenameList methods.

Disconnect Method

See Also   Error Codes   Frequently Asked Questions

Description

Performs a normal disconnect from the FTP server.

Syntax

*object.***Disconnect**

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

Part	**Description**
object	Required. An FTP control.

Remarks

When connected to an FTP server the Disconnect method simply disconnects.   Upon disconnection, the FTP control fires the Connected event.

DisablePasv Property

Frequently Asked Questions

Description

Used to disable use of PASV command with servers that don't implement it correctly.

Syntax

*object.***DisablePasv** [= *boolean*]

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

Part	**Description**
object	An FTP control.
boolean	A boolean expression that determines if the FTP control uses the PASV command.

Remarks

The FTP protocol requires that data transfers are preceded by either a PORT or PASV command.   The PASV command must be used with servers that are behind firewalls. However, some server/firewall combinations are not responding to the PASV command as required by the applicable RFCs.   This causes the control to timeout when initiating data transfers.

The DisablePasv property provides a way to deal with such servers.   When DisablePasv is set to True the PASV command will not be used and the PORT command is used instead.

The recommended way to use the DisablePasv property is to first attempt a data transfer with DisablePasv set to False.   If the transfer fails because the server fails to respond correctly then an error 10004 will be returned when Blocking = True, and AsyncError will be fired with an error code of 20605 when Blocking = False.

At this point, unless you have an unusually long timeout value set, the server will be "hung" and you must disconnect from the server, set DisablePasv to True and then reconnect, after which data transfers will probably work.   If the server still fails to transfer data after following this procedure, then it's completly broken.

The following code shows how you might handle the situation:

```
Private Sub cmdDirList_Click()
    ClearListDir
    MainForm.Ftp1.Pattern = txtPattern.Text
    On Error Resume Next
    MainForm.Ftp1.Action = FtpActionGetDirList
    If (Err = 10004) Then
        MainForm.Ftp1.Timeout = 5
        MainForm.Ftp1.Action = FtpActionDisconnect
        MainForm.Ftp1.Timeout = 30
        On Error GoTo 0
        MainForm.Ftp1.Action = FtpActionConnect
        MainForm.Ftp1.DisablePasv = True
        MainForm.Ftp1.Action = FtpActionGetDirList
    End If
End Sub
```

The timeout value is set to 5 seconds since the server is most likely hung at this point and there's no sense in waiting a long time for the control to timeout again.

Note that you will be able to determine whether the DisablePasv property must be set to True when you attempt the first data transfer.   The error will occur with file transfers too, so you need to be sure to handle the situation in all cases where a data transfer may be the first operation you attempt with a given server.

Data Type

Integer (boolean)

Done Event

See Also		Error Codes		Frequently Asked Questions

Description

Fired when an asynchronous operation completes.

Syntax

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

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

Part	**Description**
object	An FTP 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

Destination file for file-oriented methods.

Syntax

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

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

Part	**Description**
object	An FTP control.
filename	A string expression that specifies the destination (resulting) filename in a file-oriented method.

Remarks

The GetFile, PutFile, and Rename methods all require both a source and destination filename.   See each command for details regarding the use of this property.

Data Type

String

GetCurrentDir Method

See Also		Error Codes		Frequently Asked Questions

Description

Gets the current directory on the FTP server.

Syntax

*object.***GetCurrentDir**

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

Part	**Description**
object	Required. An FTP control.

Remarks

Retrieves the FTP server's current directory.   When the command completes, the Directory property will contain the server's response.

GetDirList Method

See Also   Error Codes   Frequently Asked Questions

Description

Retrieves a verbose directory listing from the FTP server.

Syntax

*object.***GetDirList** *pattern*

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

Part	**Description**
object	Required. An FTP control.
pattern	Optional. A string expression holding the search pattern (including wildcards) for the directory listing. If not specified, the Pattern property is used.

Remarks

Requests a verbose listing of the files in the FTP server's current directory. The exact format of the data returned may vary depending upon the system type. For each line of the directory listing, the DirItem event is fired.

GetFile Method

See Also		Error Codes		Frequently Asked Questions

Description

Retrieves a file from the FTP server.

Syntax

*object.***GetFile** *srcfilename*, *dstfilename*

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

Part	**Description**
object	Required. An FTP control.
srcfilename	Optional. A string expression holding the source filename. If not specified, the SrcFilename property is used.
dstfilename	Optional. A string expression specifying the destination filename. If not specified, the DstFilename property is used.

Remarks

Copies data from the remote file named by the *srcfilename* parameter to the local file named by the *dstfilename* parameter.   If the destination file already exists, its contents are replaced; otherwise it is created

GetFilenameList Method

See Also Error Codes Frequently Asked Questions

Description

Retrieves a terse directory listing from the FTP server.

Syntax

*object.***GetFilenameList** *pattern*

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

Part	**Description**
object	Required. An FTP control.
pattern	Optional. A string expression holding the search pattern (including wildcards) for the directory listing. If not specified, the Pattern property is used.

Remarks

Requests a name-only listing of the files in the FTP server's current directory. For each line of the directory listing, the DirItem event is fired.

Host Property

See Also Error Codes Frequently Asked Questions

Description

Name or IP address of the FTP host.

Syntax

*object.***Host** [= *host*]

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

Part	**Description**
object	An FTP control.
host	A string expression specifying the FTP host (either as a host name or IP address).

Remarks

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

Data Type

String

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	An FTP 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 which will fall into one of the following groups:

Values	**Description**
1-481	Standard VB error codes
10000-19999	Errors returned by WinSock. (see Error Codes)
20200-20299	Results from successfully completed commands.
20400-20499	Potentially recoverable errors reported by the server.
20500-20599	Errors reported by the server.
20600-20699	Errors unique to the control

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

Data Type

Integer

LogonName Property

See Also Error Codes Frequently Asked Questions

Description

User name on the server.

Syntax

*object.***LogonName** [= *name*]

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

Part	**Description**
object	An FTP control.
name	A string expression that holds a user name for logging on to the FTP server.

Remarks

FTP servers require that you supply a user name and password in order to connect to them.

Many FTP servers support logons by anonymous users, in which case the LogonName property should be set to "anonymous" and the LogonPassword property to the user's e-mail address, prior to invoking the Connect method.

Data Type

String

LogonPassword Property

See Also Error Codes Frequently Asked Questions

Description

Password to use when connecting to the FTP server.

Syntax

*object.***LogonPassword** [= *password*]

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

Part	**Description**
object	An FTP control.
password	A string expression that holds the password to use when logging on to an FTP server.

Remarks

FTP servers require a username and password.   This property is sent to the FTP server when it asks for a password.

Many FTP servers support anonymous user logons in which case the LogonName property should be set to "anonymous" and the LogonPassword property should be set to the user's e-mail address.

Data Type

String

ParentDir Method

See Also Error Codes Frequently Asked Questions

Description

Changes to parent directory on FTP server.

Syntax

*object.***ParentDir**

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

Part	**Description**
object	Required. An FTP control.

Remarks

Changes the FTP server's current directory to its parent directory.

Pattern Property

See Also   Error Codes   Frequently Asked Questions

Description

Wildcard pattern for directory operations.

Syntax

*object.***Pattern** [= *pattern*]

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

Part	**Description**
object	An FTP control.
pattern	A string expression that holds the filename/wildcard pattern for file-list-oriented methods.

Remarks

GetDirList and GetFilenameList permit the use of a wildcard pattern for selecting which files to list.

Setting the Pattern property appropriately (i.e., "S*" to list all files beginning with an upper-case S) prior to invoking either of those methods will restrict the list of files returned.

Data Type

String

Port Property

See Also   Error Codes   Frequently Asked Questions

Description

Port number of the FTP server.

Syntax

*object.***Port** [= *port*]

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

Part	**Description**
object	An FTP control.
port	An integer that specifies the port number to use to connect to the FTP server.

Remarks

This property specifies the port number to use when connecting (see the Connect method). This property defaults to 21, and should normally be left at the default.

This property is used when a proxy server, firewall, etc. requires that the user connect to the FTP server using a different port.

Data Type

Integer

Progress Event

See Also   Error Codes   Frequently Asked Questions

Description

Fired as lengthy operations progress.

Syntax

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

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

Part	**Description**
object	An FTP control.
index	An integer that identifies a control if it's in a control array.
bytes	A long integer that holds the number of bytes sent to / received from the FTP server.

Remarks

During file transfers the Progress event is periodically fired.   The *bytes* argument lets you know how many bytes have been transferred as of the time the Progress event was fired.

PutFile Method

See Also   Error Codes   Frequently Asked Questions

Description

Sends a file to the FTP server.

Syntax

*object.***PutFile** *srcfilename*, *dstfilename*

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

Part	**Description**
object	Required. An FTP control.
srcfilename	Optional. A string expression holding the source filename. If not specified, the SrcFilename property is used.
dstfilename	Optional. A string expression specifying the destination filename. If not specified, the DstFilename property is used.

Remarks

Copies data from the local file named by the *srcfilename* parameter to the FTP server's file named by the *dstfilename* parameter. If *dstfilename* already exists on the server, its contents are replaced. If *dstfilename* does not exist, it is created.

Some servers may require you to use the Allocate method prior to starting a file transfer.

PutUniqueFile Method

See Also Error Codes Frequently Asked Questions

Description

Transfers a file to the FTP server using a unique destination file name.

Syntax

*object.***PutUniqueFile** *srcfilename*

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

Part	**Description**
object	Required. An FTP control.
srcfilename	Optional. A string expression holding the source filename. If not specified, the SrcFilename property is used.

Remarks

Copies data from the local file named by the *srcfilename* parameter to a uniquely named file on the FTP server. The FTP server chooses the name of the destination file such that it is unique in the current directory.

Some servers may require you to use the Allocate method prior to starting a file transfer.

Quote Method

See Also Error Codes Frequently Asked Questions

Description

Sends a command string to the server.

Syntax

*object.***Quote** *command*

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

Part	**Description**
object	Required. An FTP control.
command	Optional. A string expression containing data to send to the FTP server. If this is omitted, the FTP control uses the data in the QuoteCmd property.

Remarks

Used to write a command string to the server. This method is useful for sending non-standard commands or for invoking some of the more obscure commands specified in RFC 959. After the server finishes responding, the Done event is fired if the control is in non-blocking mode. Upon completion of the Quote command, the ASCII response returned by the server can be read from the ReadData property.

Server responses consist of one or more lines of ASCII text. The first line of the response always contains a three digit code and some "user-friendly" message (i.e., ddd Ok, I'll let you do that). If the response contains more than one line, the three digits will be followed by a "-" character. Any number of lines may follow a ddd- response. The final line of the response will contain the same three digit code as the first line, but there will be no "-" character. In other words, if the first three digits of the first line of the response are not followed by a dash then the response consists of a single line only. Otherwise, the last line of the response will begin with three digits with no following dash character.

For most purposes only the first digit of the three-digit code is significant. The first digits of the three-digit codes have the following meanings:

Value	**Description**
1	Positive Preliminary Reply The requested action is being initiated; expect another reply before proceeding with a new command.
2	Positive Completion Reply The requested action has been successfully completed.
3	Positive Intermediate Reply The command has been accepted, but the requested action is being held in abeyance, pending receipt by the server of further information from the client.
4	Transient Negative Reply The command was not accepted, but the error condition is temporary and the action may be requested again.
5	Permanent Negative Completion Reply The command was not accepted and the requested action did not take place.

RFC 959 goes into greater detail regarding response codes and you would do well to read the RFC before trying to use the Quote command.

Also, note that some commands you might send with the Quote command (such as list) will attempt to initiate a data-transfer. It is possible, using a socket control and Quote commands with the FTP control, to properly set up and handle a data-transfer type command, but this is far from simple and requires intimate familiarity with sockets and RFC 959. For commands that require a data transfer, your best bet is to use the FTP control's methods and properties whenever possible.

The FTP control implements RFC recommendations for handling firewalls.   However, in cases where non-standard firewall implementations are causing a problem, the Quote command and a socket control for data-transfers provide a difficult, but doable, solution.

QuoteCmd Property

See Also   Error Codes   Frequently Asked Questions

Description

Command sent to the server with the Quote method.

Syntax

*object.***QuoteCmd** [= *quote*]

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

Part	**Description**
object	An FTP control.
quote	A string expression that specifies the command to send to the FTP server.

Remarks

You can use the FTP control's Quote method (Action = FtpActionQuote) to send arbitrary commands to an FTP server.   This is often used to invoke a server's SITE command (the site command returns a list of site-specific commands implemented by the server), or to invoke a site-specific command from the list returned by the server.

See the Quote method for further information.

Data Type

String

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. An FTP control.

Remarks

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	An FTP control.

Remarks

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

Reinitialize Method

See Also Error Codes Frequently Asked Questions

Description

Ends, then re-starts the user's FTP session.

Syntax

*object.***Reinitialize** *username*, *password*, *account*

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

Part	**Description**
object	Required. An FTP control.
username	Optional. A string expression holding the user's name. If not specified, the LogonName property is used.
password	Optional. A string expression specifying the user's password. If not specified, the LogonPassword property is used.
account	Optional. A string expression specifying the user's account name. If not specified, the Account property is used.

Remarks

Terminates the current user's FTP session and starts a new session using the LogonName, LogonPassword, and Account properties as they are used by the Connect method.

Rename Method

See Also Error Codes Frequently Asked Questions

Description

Renames a file on the FTP server.

Syntax

*object.***Rename** *srcfilename*, *dstfilename*

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

Part	**Description**
object	Required. An FTP control.
srcfilename	Optional. A string expression holding the source filename. If not specified, the SrcFilename property is used.
dstfilename	Optional. A string expression specifying the destination filename. If not specified, the DstFilename property is used.

Remarks

Renames the remote file named by the *srcfilename* parameter to the name contained in the *dstfilename* parameter.

SendType Method

See Also Error Codes Frequently Asked Questions

Description

Specifies the file transfer mode.

Syntax

*object.***SendType** *type*

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

Part	**Description**
object	Required. An FTP control.
type	Optional. An integer expression holding the desired send type/method. If not specified, the Type property is used.

Remarks

Sets the server's transfer mode to either ASCII (Type = 0) or Binary (Type = 1).

SrcFilename Property

See Also Error Codes Frequently Asked Questions

Description

Source file for file-oriented methods.

Syntax

*object.***SrcFilename** [= *filename*]

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

Part	**Description**
object	An FTP control.
filename	A string expression that specifies the source filename to use with a file-oriented method.

Remarks

The GetFile, PutFile, and Rename methods all require both a source and destination filename. See each command for details regarding the use of this property.

Data Type

String

State Property

See Also Error Codes Frequently Asked Questions

Description

The current state of the control.

Syntax

*object.***State**

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

Part	**Description**
object	An FTP control.

Remarks

This property may take on many values while executing commands. The only values of use in your program are 0 (Disconnected) and 1 (Connected). You must not rely on the meaning of any other values which this property may contain. When State is 0 then the only valid method is Connect. No other method may be executed while State is not equal to 1 (Connected).

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

Data Type

Integer

Timeout Property

See Also   Error Codes   Frequently Asked Questions

Description

Time in seconds to wait for an operation to complete.

Syntax

*object.***Timeout** [= *duration*]

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

Part	**Description**
object	An FTP control.
duration	An integer that determines how long, in seconds, the control will wait. Setting to zero means that the control will wait forever.

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

Type Property

See Also Error Codes Frequently Asked Questions

Description

Type of data to transfer.

Syntax

*object.*__Type__ [= *type*]

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

Part	**Description**
object	An FTP control.
type	An integer that specifies the method of data transfer (ASCII or binary).

Remarks

FTP servers require that you specify the type of data you expect to be in the files you're transferring back and forth. The Type property has two possible values:

Value	**Description**
0	ASCII
1	Binary (default)

FTP servers default to ASCII transfers. Unless you're sure that you will be transferring ASCII-only files, it is best to switch Type to 1 (Binary) prior to any file transfers.

This property defaults to binary transfers (1).

Data Type

Integer (enumerated)

Version Property

Error Codes Frequently Asked Questions

Description

Shows the version of the control.

Syntax

*object.***Version**

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

Part	**Description**
object	An FTP 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** *writedata*

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

Part	**Description**
object	Required. An FTP control.
writedata	Optional. A string expression containing data to send to the FTP server. If this is omitted, the FTP control uses the data in the WriteData property.

Remarks

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	An FTP control.
string	A string expression that holds data to send to the FTP server.

Remarks

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

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.