Help for FTP/X

Properties Events Methods Interfaces Frequently Asked Questions

How To Buy This Software

Order Form

Getting Custom Controls Written

Licensing Information

Description

Mabry's FTP/X ActiveX control provides easy, high-level access to the complete FTP client protocol (RFC 959). In addition to capturing server directory listings into a string array property, the new FTP/X also makes the results available as an ADO Recordset providing easy access to the various fields that a server returns. The FTP/X also has powerful built-in features to support debugging and non-standard servers using the Quote method.

The 32-bit FTP/X comes as both an ActiveX control (OCX) and a COM object (DLL), so you can use it in nearly any modern programming environment. As a light ActiveX control/COM Object, FTP/X doesn't require any bulky MFC DLLs to run. Your web-based applications (VBScript, ASP, etc.) will download faster, your installation package will be smaller, and you'll have no problems with different versions of DLLs on your users' systems. And, because it's both an ActiveX control and COM Object, FTP/X will run in Visual Basic, Visual FoxPro, Internet Explorer, Delphi and any other environment that supports either ActiveX controls or COM objects.

A common problem with FTP servers is that there is no official standard when it comes to the format of directory listings. The exact format of the data returned by the server may vary depending upon the system type. The FTP/X control solves this programmer's headache by automatically parsing the data and making it available to your application as a familiar ADO Recordset object. If the server's format is not suitable for the control's parser, a new DirItemPattern property can be used to specify how the fields are to be parsed. For those who would rather handle the data themselves, the data is also presented in its original format via a string array property.

Both the FTP/X ActiveX control and COM Object use the normal event-driven programming model or Fast Notifications. Fast Notifications allow your program to quickly receive events through simple functions, rather than going through the overhead to fire an event. This reduces the amount of time required to handle an event.

FTP/X supports non-blocking (asynchronous) mode as well as pseudo-blocking and true blocking modes, to meet most any programming requirement. FTP/X is firewall friendly complying with the Firewall-Friendly FTP Specification (RFC 1579). From behind most firewalls, your applications will be able to communicate with external FTP servers.

Mabry's FTP/X ActiveX control requires no user input to run. The FTP/X control can run completely in the background under program control. This allows your program to take care of file transfers while the user performs other tasks.

File Name

FTPX.OCX, FTPX.DLL

ActiveX / OCX Object Name

Mabry.FtpX.1

ActiveX Compatibility

VB 4.0 (32-bit), 5.0 and 6.0

ActiveX Built With

Microsoft Visual C++ v6

ActiveX - Required DLLs

None. This is a light ActiveX control that requires no extra DLLs to operate. This means that your installation packages will be smaller. And, your web pages will load faster, since there are no extra DLLs to download.

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/X Properties

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

Close

FTP/X Events

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

*__Connected__ Event

*__Debug__ Event

*__DirItem__ Event

*__Disconnected__ Event

*__Done__ Event

*__Progress__ Event

*__StateChanged__ Event

Close

FTP/X Methods

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

*__Abort__ Method
*__About__ Method
*__Allocate__ Method
*__Append__ Method
*__ChangeDir__ Method
*__Connect__ Method
*__CreateDir__ Method
*__Delete__ Method
*__DeleteDir__ Method
*__Disconnect__ Method
*__GetCurrentDir__ Method
*__GetDirList__ Method
*__GetFile__ Method
*__GetFilenameList__ Method
*__ParentDir__ Method
*__PutFile__ Method
*__PutUniqueFile__ Method
*__Quote__ Method
*__Reinitialize__ Method
*__Rename__ Method
*__SendType__ Method

Close

FTP/X Interfaces

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

*__IFtpXCtlNotify__ Interface

*__IFtpXObjNotify__ Interface

Close

FTP/X IFtpXCtlNotify Interface Events

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

*__Connected__ Event

*__Debug__ Event

*__DirItem__ Event

*__Disconnected__ Event

*__Done__ Event

*__Progress__ Event

*__StateChanged__ Event

Close

FTP/X IFtpXObjNotify Interface Events

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

*__Connected__ Event

*__Debug__ Event

*__DirItem__ Event

*__Disconnected__ Event

*__Done__ Event

*__Progress__ Event

*__StateChanged__ Event

Close

Frequently Asked Questions

General Questions

How do I delete a file using your FTP control?

Frequently Asked Questions

Set the DstFilename property to the name of the file on the server and then invoke the Delete method (or Action):

```
mFTP1.DstFilename = "STAR.TMP"
mFTP1.Delete
```

or

```
mFTP1.DstFilename = "STAR.TMP"
mFTP1.Action = FtpActionDelete
```

I'm having troubles using the FTP control in Visual FoxPro. Any tips?

Frequently Asked Questions

Sure.

1. There is a sample VFP project available at the Mabry website under the Sample Code section.

2. Make sure you are using VFP 5.0 or later. The controls probably won't work in earlier versions.

3. For some reason, blocking mode doesn't work in VFP. The control must be used in non-blocking mode.

4. Objects that are created using the Create method do not receive events. Since non-blocking mode requires the Done event, you cannot instantiate the control using the Create method. You must place the control on a form by using the VFP OLEContainer control.

5. VFP can miss events when many are fired in succession (such as the FTP control's DirItem event). Make certain that AutoYield is false and your application doesn't do anything other than handle the events in a multi-event scenerio.

6. If using one app to launch other apps, we've seen VFP get "confused" as to which form "ThisForm" is referencing. In these cases, the main form must contain a copy of each object.

7. Some properties and events are named the same (Debug, in particular). Apparently, VFP can't deal with this, but according to the COM rules, we cannot change the name because then the control is not backwards compatible. This can inhibit debugging.

How To Buy This Software

CREDITS

FTP/X 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.
503 316th Street Northwest
Stanwood, WA 98292

Sales: 1-800-99-MABRY (U.S. Only)
Voice: 360-629-9278
Fax: 360-629-9278
Web: http://www.mabry.com

COST

The price of FTP/X (control only) is US$50 (US$55 for International orders). The cost of FTP/X and the C/C++ source code (of the control itself) is US$199 (US$204 for International orders).

Prices are subject to change without notice.

Printed manuals are available at US$12.50 per copy.

PART NUMBERS

The product number for FTP/X (control only) is 15749.

The product number for and the C/C++ source code (of the control itself) is 15750.

DELIVERY METHODS

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

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

US$10.00	US Priority Mail
US$15.00	Airborne Express 2nd Day (US deliveries only)
US$20.00	Airborne Express Overnight (US deliveries only)
US$20.00	Global Priority Mail (Int'l deliveries only; Western Europe, Pacific Rim and Canada only)
US$45.00	International Airborne Express (Int'l deliveries only)

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). We will 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.

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, Discover, NOVUS), 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$25.00 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/X Order Form

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

Mail this form to:

Mabry Software, Inc.
503 316th Street Northwest
Stanwood, WA 98292

Phone: 360-629-9278
Fax: 360-629-9278
Internet: mabry@mabry.com
Web: www.mabry.com

Where did you get this copy of FTP/X?

_____

Name: _____

Ship to: _____

_____

_____

_____

Phone: _____

Fax: _____

E-Mail: _____

Credit Card #: _____ exp. _____

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

qty ordered ____ REGISTRATION
$50.00 ($55.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $10.00 per order for shipping and handling. Add $12.50 per printed manual.

qty ordered ____ SOURCE CODE AND REGISTRATION
$199.00 ($204.00 international). Check or money order in U.S. currency drawn on a U.S. bank. Add $10.00 per order for shipping and handling. Add $12.50 per printed manual.

Close Copy Print

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
BlockingMode Property
Done Event
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property

See Also

Version Property

See Also

Connect Method
LogonName Property
LogonPassword Property
Reinitialize Method

See Also

AllocBytes Property
Append Method
Done Event
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastError Property
LastMethod Property
PutFile Method
PutUniqueFile Method

See Also

Allocate Method

PutFile Method

See Also

- **Allocate** Method
- **Done** Event
- **DstFilename** Property
- **GetFile** Method
- **IFtpXCtlNotify** Done Event
- **IFtpXCtlNotify** Progress Event
- **IFtpXObjNotify** Done Event
- **IFtpXObjNotify** Progress Event
- **LastMethod** Property
- **Progress** Event
- **PutFile** Method
- **PutUniqueFile** Method
- **Rename** Method
- **SrcFilename** Property
- **Type** Property

See Also

Abort Method
BlockingMode Property
Connect Method
Done Event

See Also

Abort Method

Blocking Property

Done Event

See Also

CreateDir Method
DeleteDir Method
Directory Property
Done Event
GetCurrentDir Method
GetDirList Method
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property
ParentDir Method

See Also

- **Account** Property
- **Blocking** Property
- **Connected** Event
- **DisablePASV** Property
- **Disconnect** Method
- **Done** Event
- **Host** Property
- **IFtpXCtlNotify** Connected Event
- **IFtpXCtlNotify** Done Event
- **IFtpXObjNotify** Connected Event
- **IFtpXObjNotify** Done Event
- **LastMethod** Property
- **LibraryName** Property
- **LogonName** Property
- **LogonPassword** Property
- **Port** Property
- **Reinitialize** Method
- **State** Property

See Also

- **Connect** Method
- **Disconnect** Method
- **Done** Event
- **Reinitialize** Method
- **State** Property
- **StateChanged** Event
- **StateString** Property

See Also

ChangeDir Method
DeleteDir Method
Directory Property
Done Event
GetCurrentDir Method
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property
ParentDir Method

See Also

DebugMode Property

PutUniqueFile Method

See Also

Debug Event

IFtpXCtlNotify Debug Event

IFtpXObjNotify Debug Event

See Also

DeleteDir Method
Done Event
DstFilename Property
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property
SrcFilename Property

See Also

- **ChangeDir** Method
- **CreateDir** Method
- **Delete** Method
- **Directory** Property
- **Done** Event
- **GetCurrentDir** Method
- **IFtpXCtlNotify** Done Event
- **IFtpXObjNotify** Done Event
- **LastMethod** Property

See Also

ChangeDir Method
CreateDir Method
DeleteDir Method
GetCurrentDir Method

See Also

DirItem Event
DirItemPattern Property
DirItems Property
GetDirList Method
GetFilenameList Method
IFtpXCtlNotify DirItem Event
IFtpXObjNotify DirItem Event

See Also

DirItem Property
DirItemPattern Property
DirItems Property
GetDirList Method
GetFilenameList Method
Recordset Property

See Also

DirItem Event

DirItem Property

GetDirList Method

IFtpXCtlNotify DirItem Event

IFtpXObjNotify DirItem Event

Recordset Property

See Also

DirItem Event
DirItem Property
GetDirList Method
GetFilenameList Method
IFtpXCtlNotify DirItem Event
IFtpXObjNotify DirItem Event

See Also

Connect Method

Disconnect Method

See Also

Connect Method
Connected Event
DisablePasv Property
Disconnected Event
Done Event
IFtpXCtlNotify Disconnected Event
IFtpXCtlNotify Done Event
IFtpXObjNotify Disconnected Event
IFtpXObjNotify Done Event
LastMethod Property
Reinitialize Method

See Also

Disconnect Method
Done Event
State Property
StateChanged Event
StateString Property

See Also

- **Abort** Method
- **Allocate** Method
- **Append** Method
- **Blocking** Property
- **BlockingMode** Property
- **ChangeDir** Method
- **Connect** Method
- **Connected** Event
- **CreateDir** Method
- **Delete** Method
- **DeleteDir** Method
- **Disconnect** Method
- **Disconnected** Event
- **GetCurrentDir** Method
- **GetDirList** Method
- **GetFile** Method
- **GetFilenameList** Method
- **LastError** Property
- **LastErrorString** Property
- **LastMethod** Property
- **ParentDir** Method
- **PutFile** Method
- **PutUniqueFile** Method
- **Quote** Method
- **QuoteCmd** Property
- **ReadData** Property
- **Reinitialize** Method
- **Rename** Method
- **SendType** Method
- **Timeout** Property

See Also

Append Method
Delete Method
GetFile Method
PutFile Method
Rename Method
SrcFilename Property

See Also

ChangeDir Method
CreateDir Method
DeleteDir Method
Directory Property
Done Event
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property
ParentDir Method

See Also

ChangeDir Method
DirItem Event
DirItem Property
DirItemPattern Property
DirItems Property
Done Event
GetFilenameList Method
IFtpXCtlNotify DirItem Event
IFtpXCtlNotify Done Event
IFtpXObjNotify DirItem Event
IFtpXObjNotify Done Event
LastMethod Property
ParentDir Method
Pattern Property
Recordset Property

See Also

Append Method
Done Event
DstFilename Property
IFtpXCtlNotify Done Event
IFtpXCtlNotify Progress Event
IFtpXObjNotify Done Event
IFtpXObjNotify Progress Event
LastMethod Property
Progress Event
PutFile Method
SendType Method
SrcFilename Property
Type Property

See Also

DirItem Event
DirItem Property
DirItems Property
Done Event
GetDirList Method
IFtpXCtlNotify DirItem Event
IFtpXCtlNotify Done Event
IFtpXObjNotify DirItem Event
IFtpXObjNotify Done Event
LastMethod Property
Pattern Property

See Also

Connect Method

Port Property

Reinitialize Method

See Also

- **IFtpXCtlNotify** Connected Event
- **IFtpXCtlNotify** Debug Event
- **IFtpXCtlNotify** DirItem Event
- **IFtpXCtlNotify** Disconnected Event
- **IFtpXCtlNotify** Done Event
- **IFtpXCtlNotify** Progress Event
- **IFtpXCtlNotify** StateChanged Event
- **NotificationObject** Property

See Also

IFtpXObjNotify Connected Event
IFtpXObjNotify Debug Event
IFtpXObjNotify DirItem Event
IFtpXObjNotify Disconnected Event
IFtpXObjNotify Done Event
IFtpXObjNotify Progress Event
IFtpXObjNotify StateChanged Event
NotificationObject Property

See Also

Allocate Method
Done Event
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastErrorString Property

See Also

Done Event

IFtpXCtlNotify Done Event

IFtpXObjNotify Done Event

LastError Property

See Also

Abort Method
Allocate Method
Append Method
ChangeDir Method
Connect Method
CreateDir Method
Delete Method
DeleteDir Method
Disconnect Method
Done Event
GetCurrentDir Method
GetDirList Method
GetFile Method
GetFilenameList Method
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
ParentDir Method
PutFile Method
PutUniqueFile Method
Quote Method
Reinitialize Method
Rename Method
SendType Method

See Also

Connect Method

See Also

Account Property
Connect Method
LogonPassword Property
Reinitialize Method

See Also

- **Account** Property
- **Connect** Method
- **LogonName** Property
- **Reinitialize** Method

See Also

IFtpXCtlNotify Connected Event
IFtpXCtlNotify Debug Event
IFtpXCtlNotify Interface
IFtpXObjNotify Connected Event
IFtpXObjNotify Debug Event
IFtpXObjNotify Interface

See Also

ChangeDir Method
CreateDir Method
Done Event
GetCurrentDir Method
GetDirList Method
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property

See Also

GetDirList Method

GetFilenameList Method

See Also

Connect Method

Host Property

See Also

Append Method
GetFile Method
PutFile Method
PutUniqueFile Method

See Also

- **Allocate** Method
- **AllocBytes** Property
- **Append** Method
- **Done** Event
- **DstFilename** Property
- **GetFile** Method
- **IFtpXCtlNotify** Done Event
- **IFtpXCtlNotify** Progress Event
- **IFtpXObjNotify** Done Event
- **IFtpXObjNotify** Progress Event
- **LastMethod** Property
- **Progress** Event
- **PutUniqueFile** Method
- **Rename** Method
- **SendType** Method
- **SrcFilename** Property
- **Type** Property

See Also

- **Allocate** Method
- **Append** Method
- **Debug** Event
- **Done** Event
- **IFtpXCtlNotify** Done Event
- **IFtpXCtlNotify** Progress Event
- **IFtpXObjNotify** Done Event
- **IFtpXObjNotify** Progress Event
- **LastMethod** Property
- **Progress** Event
- **PutFile** Method
- **SendType** Method
- **SrcFilename** Property
- **Type** Property

See Also

Done Event
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property
QuoteCmd Property
ReadData Property

See Also

Done Event

Quote Method

ReadData Property

See Also

Done Event

Quote Method

QuoteCmd Property

See Also

DirItem Event
DirItemPattern Property
GetDirList Method
IFtpXCtlNotify DirItem Event
IFtpXObjNotify DirItem Event

See Also

- **Account** Property
- **Connect** Method
- **Connected** Event
- **Disconnect** Method
- **Done** Event
- **Host** Property
- **IFtpXCtlNotify** Connected Event
- **IFtpXCtlNotify** Done Event
- **IFtpXObjNotify** Connected Event
- **IFtpXObjNotify** Done Event
- **LastMethod** Property
- **LogonName** Property
- **LogonPassword** Property

See Also

Append Method
Done Event
DstFilename Property
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property
PutFile Method
SendType Method
SrcFilename Property

See Also

Done Event
GetFile Method
IFtpXCtlNotify Done Event
IFtpXObjNotify Done Event
LastMethod Property
PutFile Method
PutUniqueFile Method
Rename Method
Type Property

See Also

Append Method
Delete Method
DstFilename Property
GetFile Method
PutFile Method
PutUniqueFile Method
Rename Method

See Also

Connect Method
Connected Event
Disconnected Event
IFtpXCtlNotify Connected Event
IFtpXCtlNotify Disconnected Event
IFtpXCtlNotify StateChanged Event
IFtpXObjNotify Connected Event
IFtpXObjNotify Disconnected Event
IFtpXObjNotify StateChanged Event
StateChanged Event
StateString Property

See Also

Connected Event
Disconnected Event
State Property
StateString Property

See Also

Connected Event
Disconnected Event
IFtpXCtlNotify Connected Event
IFtpXCtlNotify Disconnected Event
IFtpXCtlNotify StateChanged Event
IFtpXObjNotify Connected Event
IFtpXObjNotify Disconnected Event
IFtpXObjNotify StateChanged Event
State Property
StateChanged Event

See Also

Done Event

See Also

Append Method
GetFile Method
PutFile Method
PutUniqueFile Method
SendType Method

See Also

About Method

See Also

Connect Method
IFtpXCtlNotify Done Event
IFtpXCtlNotify Interface
IFtpXCtlNotify StateChanged Event
NotificationObject Property
Reinitialize Method
State Property
StateString Property

See Also

DebugMode Property

IFtpXCtlNotify Interface

NotificationObject Property

See Also

DirItem Property
DirItemPattern Property
DirItems Property
GetDirList Method
GetFilenameList Method
IFtpXCtlNotify Interface
Recordset Property

See Also

Disconnect Method
IFtpXCtlNotify Done Event
IFtpXCtlNotify Interface
IFtpXCtlNotify StateChanged Event
State Property
StateString Property

See Also

- **Abort** Method
- **Allocate** Method
- **Append** Method
- **ChangeDir** Method
- **Connect** Method
- **CreateDir** Method
- **Delete** Method
- **DeleteDir** Method
- **Disconnect** Method
- **GetCurrentDir** Method
- **GetDirList** Method
- **GetFile** Method
- **GetFilenameList** Method
- **IFtpXCtlNotify** Connected Event
- **IFtpXCtlNotify** Disconnected Event
- **IFtpXCtlNotify** Interface
- **LastError** Property
- **LastErrorString** Property
- **LastMethod** Property
- **ParentDir** Method
- **PutFile** Method
- **PutUniqueFile** Method
- **Quote** Method
- **Reinitialize** Method
- **Rename** Method
- **SendType** Method

See Also

Append Method
GetFile Method
IFtpXCtlNotify Interface
PutFile Method
PutUniqueFile Method

See Also

IFtpXCtlNotify Connected Event
IFtpXCtlNotify Disconnected Event
IFtpXCtlNotify Interface
State Property
StateString Property

See Also

- **Connect** Method
- **IFtpXObjNotify** Done Event
- **IFtpXObjNotify** Interface
- **IFtpXObjNotify** StateChanged Event
- **NotificationObject** Property
- **Reinitialize** Method
- **State** Property
- **StateString** Property

See Also

DebugMode Property

IFtpXObjNotify Interface

NotificationObject Property

See Also

DirItem Property
DirItemPattern Property
DirItems Property
GetDirList Method
GetFilenameList Method
IFtpXObjNotify Interface
Recordset Property

See Also

Disconnect Method
IFtpXObjNotify Done Event
IFtpXObjNotify Interface
IFtpXObjNotify StateChanged Event
State Property
StateString Property

See Also

- **Abort** Method
- **Allocate** Method
- **Append** Method
- **ChangeDir** Method
- **Connect** Method
- **CreateDir** Method
- **Delete** Method
- **DeleteDir** Method
- **Disconnect** Method
- **GetCurrentDir** Method
- **GetDirList** Method
- **GetFile** Method
- **GetFilenameList** Method
- **IFtpXObjNotify** Connected Event
- **IFtpXObjNotify** Disconnected Event
- **IFtpXObjNotify** Interface
- **LastError** Property
- **LastErrorString** Property
- **LastMethod** Property
- **ParentDir** Method
- **PutFile** Method
- **PutUniqueFile** Method
- **Quote** Method
- **Reinitialize** Method
- **Rename** Method
- **SendType** Method

See Also

Append Method
GetFile Method
IFtpXObjNotify Interface
PutFile Method
PutUniqueFile Method

See Also

IFtpXObjNotify Connected Event
IFtpXObjNotify Disconnected Event
IFtpXObjNotify Interface
State Property
StateString Property

Abort Method

See Also Frequently Asked Questions

Description

Abort current data transfer.

Syntax

object.**Abort**

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

Part	Description
object	Required. An FTP/X 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.

RFC equivalent - ABOR

About Method

See Also Frequently Asked Questions

Description

Displays About Box.

Syntax

object.**About**

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

Part	Description
object	Required. An FTP/X control.

Remarks

This method displays the About Box for the control.

Account Property

See Also Frequently Asked Questions

Description

Account name, required by some servers.

Syntax

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

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

Part	Description
object	An FTP/X control.
Account	A string expression containing the Account setting.

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

Allocate Method

See Also Frequently Asked Questions

Description

Pre-allocates space on server for file transfer.

Syntax

object.**Allocate***NumberOfBytes*

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

Part	Description
object	Required. An FTP/X control.
NumberOfBytes	Required. Optional argument containing the number of bytes to allocate.

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.

RFC equivalent - ALLO

AllocBytes Property

See Also Frequently Asked Questions

Description

Bytes to allocate when using the Allocate method .

Syntax

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

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

Part	Description
object	An FTP/X control.
AllocBytes	A long integer containg the number of bytes to allocate.

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

Long

Append Method

See Also Frequently Asked Questions

Description

Appends the source (local) file to the destination (server) file.

Syntax

object.**Append***SrcFilename*,*DstFilename*

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

Part	Description
object	Required. An FTP/X control.
SrcFilename	Optional. A string expression containing the source (local) filename to send.
DstFilename	Optional. A string expression containing the destination (server) file to be appended.

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 on the server, it will be created.

If *srcfilename* and *dstfilename* are not specified, the SrcFilename Property and DstFilename Property values will be used.

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

RFC equivalent - APPE

BlockingMode Property

See Also Frequently Asked Questions

Description

Determines message processing when Blocking is True.

Syntax

object.**BlockingMode** [= *BlockingMode*]

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

Part	Description
object	An FTP/X control.
BlockingMode	Integer that indicates the type of blocking mode to use.

Remarks

There are two different block modes to choose from, FtpTrueBlocking (0) and FtpPseudoBlocking (1). If you have Blocking enabled and FtpTrueBlocking mode selected, then blocking calls will process only WM_PAINT messages for your program. All other messages will be ignored.

If you choose FtpPseudoBlocking mode then all of your program's messages will be processed, including mouse, keyboard, and so on. PseudoBlocking allows full UI interactivity while blocking operations are in progress, but also requires that you exercise considerable care to ensure that you do not call any other blocking functions while one is in progress. If you fail to handle this correctly, errors will result (which you may choose to handle with On Error Goto or similar statements).

This property supports the following values:

Constant	Value	Description
FtpTrueBlocking	0	True blocking
FtpPseudoBlocking	1	Pseudo blocking

Data Type

BlockingModesEnum

Blocking Property

See Also Frequently Asked Questions

Description

Determines if methods are blocking (synchronous) or non-blocking (asynchronous).

Syntax

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

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

Part	Description
object	An FTP/X control.
Blocking	A boolean expression that determines if the control waits for completion of an operation (True), or returns control to the program immediately (False).

Remarks

If this property is set to True, any commands using 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

Boolean

ChangeDir Method

See Also Frequently Asked Questions

Description

Changes the server's working directory.

Syntax

object.**ChangeDir***directory*

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

Part	Description
object	Required. An FTP/X 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.

(p> If the *directory* parameter is not specified, the Directory Property is used.RFC equivalent - CWD

Connect Method

See Also Frequently Asked Questions

Description

Connects to a FTP server.

Syntax

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

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

Part	Description
object	Required. An FTP/X 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

Establishes the TCP/IP connection with the server specified by the Host Property. Before executing the Connect method you must first set the Host property. If the *logonname, logonpassword, and account* parameters are not specified, the LogonName Property and the LogonPassword Property must also be set. Some servers may require an Account name and the Account Property needs to be set for those servers.

Important Note #1: 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.

Important Note #2: The DisablePASV property must not be changed while a connection is established. You must set the DisablePASV property as desired prior to establishing the connection.

RFC equivalents - USER, PASS, ACCT

IFtpXCtlNotify Interface Connected Event

See Also Frequently Asked Questions

Description

Fires after a successful Connection.

Syntax

Sub *object*_**Connected(**[*index* **As Integer**,] *FtpXControl* **As ByRef IFtpXCtl)**

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

Part	Description
object	Required. An instance of an IFtpXCtlNotify interface.
FtpXControl	A Ftp/X control.

Remarks

Fires when the State property changes to *StateConnected* (1). Note -- use caution when invoking methods from the Connected event because the IFtpXCtlNotify Done event also fires *after* the Connected event. Invoking a method from within the Connected event will cause two Done events to fire -- one for the method called from within the Connected event and one for the original Connect method.

IFtpXObjNotify Interface Connected Event

See Also Frequently Asked Questions

Description

Fires after a successful Connection.

Syntax

Sub *object*_**Connected(**[*index* **As Integer**,] *FtpXObject* **As IFtpXObj)**

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

Part	Description
object	Required. An instance of an IFtpXObjNotify interface.
FtpXObject	A Ftp/X COM Object.

Remarks

Fires when the State property changes to *StateConnected* (1). Note -- use caution when invoking methods from the Connected event because the IFtpXObjNotify Done event also fires *after* the Connected event. Invoking a method from within the Connected event will cause two Done events to fire -- one for the method called from within the Connected event and one for the original Connect method.

Connected Event

See Also Frequently Asked Questions

Description

Fires after a successful Connection.

Syntax

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

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

Part	Description
object	An FTP/X control.
index	An integer that identifies a control if it's in a control array.

Remarks

Fires when the State property changes to *StateConnected* (1). Note -- use caution when invoking methods from the Connected event because the Done event also fires *after* the Connected event. Invoking a method from within the Connected event will cause two Done events to fire -- one for the method called from within the Connected event and one for the original Connect method.

CreateDir Method

See Also Frequently Asked Questions

Description

Creates a directory on the remote system (FTP server).

Syntax

object.**CreateDir***directory*

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

Part	Description
object	Required. An FTP/X 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. If the *directory* parameter is not specified, the Directory property is used.

RFC equivalent - MKD

DebugMode Property

See Also Frequently Asked Questions

Description

Enables and disables the Debug event.

Syntax

object.**DebugMode** [= *DebugMode*]

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

Part	Description
object	An FTP/X control.
DebugMode	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

IFtpXCtlNotify Interface Debug Event

See Also Frequently Asked Questions

Description

Fired when the control has debugging information for the program.

Syntax

Sub *object*_**Debug(**[*index* **As Integer**,] *FtpXControl* **As ByRef IFtpXCtl**, *DebugMsg* **As String)**

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

Part	Description
object	Required. An instance of an IFtpXCtlNotify interface.
FtpXControl	A Ftp/X control.
DebugMsg	A string expression that holds a debugging message from the control.

Remarks

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

Debug Event

See Also Frequently Asked Questions

Description

Fired when the control has debugging information for the program.

Syntax

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

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

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

Remarks

The Debug event is enabled by setting the DebugMode property to a non-zero value (1 is the only permitted non-zero value at this time). When the DebugMode 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.

IFtpXObjNotify Interface Debug Event

See Also Frequently Asked Questions

Description

Fired when the control has debugging information for the program.

Syntax

Sub *object*_**Debug(**[*index* **As Integer**,] *FtpXObject* **As IFtpXObj**, *DebugMsg* **As String)**

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

Part	Description
object	Required. An instance of an IFtpXObjNotify interface.
FtpXObject	A Ftp/X COM Object.
DebugMsg	A string expression that holds a debugging message from the control.

Remarks

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

Delete Method

See Also Frequently Asked Questions

Description

Deletes a file from the remote system.

Syntax

object.**Delete***DstFilename*

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

Part	Description
object	Required. An FTP/X control.
DstFilename	Optional. A string expression containing the name of the file to delete from the server.

Remarks

This method deletes the file specified by *dstfilename* from the FTP server. If the *dstfilename* parameter is not specified then the DstFilename property is used.

RFC equivalent - DELE

DeleteDir Method

See Also Frequently Asked Questions

Description

Deletes a directory from the remote system.

Syntax

object.**DeleteDir***directory*

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

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

Remarks

This method deletes a directory, specified by the *directory* parameter, from the FTP server. If the *directory* parameter is not specified then the Directory property is used.

RFC equivalent - RMD

Directory Property

See Also Frequently Asked Questions

Description

Argument for directory-oriented methods.

Syntax

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

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

Part	Description
object	An FTP/X control.
Directory	A string expression containing a directory name for the FTP server.

Remarks

You can set this property to a directory name for the FTP server before using the ChangeDir method, CreateDir method, or DeleteDir method.

This property is set to the server's current directory upon completion of the GetCurrentDir method.

Data Type

String

DirItem Event

See Also Frequently Asked Questions

Description

Fires as the server returns data during the GetDirList or GetFilenameList methods.

Syntax

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

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

Part	Description
object	An FTP/X control.
index	An integer that identifies a control if it's in a control array.
Item	A string expression containing the item returned by the server.

Remarks

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

IFtpXCtlNotify Interface DirItem Event

See Also Frequently Asked Questions

Description

Fires as the server returns data during the GetDirList or GetFilenameList methods.

Syntax

Sub *object*_**DirItem(**[*index* **As Integer**,] *FtpXControl* **As ByRef IFtpXCtl**, *Item* **As String)**

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

Part	Description
object	Required. An instance of an IFtpXCtlNotify interface.
FtpXControl	A Ftp/X control.
Item	A string expression containing the item returned by the server.

Remarks

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

DirItem Property

See Also Frequently Asked Questions

Description

This property contains the data retrieved from a GetDirList or GetFilenameList method. It is a string array where each element in the array contains one item returned by the server. The index ranges from zero (0) to (DirItems - 1).

The exact format of the data returned by the server may vary depending upon the system type.

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

Syntax

object.**DirItem** [= *Index*, *DirItem*]

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

Part	Description
object	An FTP/X control.
Index	A long integer that identifies an element in the array.
DirItem	A string expression that contains the actual item as returned by the server.

Remarks

This property contains the data retrieved from a GetDirList or GetFilenameList method. It is a string array where each element in the array contains one item returned by the server. The index ranges from zero (0) to (DirItems - 1).

The exact format of the data returned by the server may vary depending upon the system type. Use the DirItemPattern property to customize the pattern if necessary.

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

Data Type

String

IFtpXObjNotify Interface DirItem Event

See Also Frequently Asked Questions

Description

Fires as the server returns data during the GetDirList or GetFilenameList methods.

Syntax

Sub *object*_**DirItem(**[*index* **As Integer**,] *FtpXObject* **As IFtpXObj**, *Item* **As String)**

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

Part	Description
object	Required. An instance of an IFtpXObjNotify interface.
FtpXObject	A Ftp/X COM Object.
Item	A string expression containing the item returned by the server.

Remarks

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

DirItemPattern Property

See Also Frequently Asked Questions

Description

Allows customization of the format for directory listings.

Syntax

object.**DirItemPattern** [= *DirItemPattern*]

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

Part	Description
object	An FTP/X control.
DirItemPattern	A string expression that specifies a custom parsing string for items returned during a GetDirList.

Remarks

The GetDirList method causes the server to return a string which is the source of the Recordset property. The exact format of the data returned by the server may vary depending upon the system type. The DirItemPattern property can be used to specify how the fields are parsed. For each directory item returned by the server the line is parsed by taking the characters from DirItemPattern one at a time, left to right, and applying the following rules:

1. If the letter in the DirItemPattern is upper-case, then copy characters from the directory item to the appropriate recordset field until either a whitespace char or end of line character is found.

2. The exception to rule 1 is when the DirItemPattern is the upper case "D" character. "D" parses the three parts of a Date (month, day, and year, such as Mar 3 1996) and concatenates them into the recordset's Date field.

3. If the DirItemPattern character is a lower-case letter, only a single character is copied into the appropriate field. Characters in the pattern which do not match a field are skipped, as is the character from the directory item.

The characters have the following meanings:

T: Filetype
P: Permissions
N: Filename
L: Link count
O:Owner
G: Group
D: Date
S: Size

A blank DirItemPattern value is the equivalent to using "tPLOGSDN". This default pattern should work for every unix or NT-based system running a standard configuration.

Data Type

String

DirItems Property

See Also Frequently Asked Questions

Description

Indicates the number of elements in the DirItem array.

Syntax

object.**DirItems** [= *DirItems*]

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

Part	Description
object	An FTP/X control.
DirItems	A long integer that holds the number of items in the DirItem array.

Remarks

DirItems holds the number of items received from the last GetDirList method or GetFilenameList method. This property tells you how many items are in the DirItem property array. This property is only valid after a GetDirList or GetFilenameList method.

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

Data Type

Long

IFtpXCtlNotify Interface Disconnected Event

See Also Frequently Asked Questions

Description

Fires after disconnecting from the FTP server.

Syntax

Sub *object*_**Disconnected(**[*index* **As Integer**,] *FtpXControl* **As ByRef IFtpXCtl)**

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

Part	Description
object	Required. An instance of an IFtpXCtlNotify interface.
FtpXControl	A Ftp/X control.

Remarks

Fires when the State property changes to *StateNotConnected* (0). Note -- the IFtpXCtlNotify Done event also fires *after* the IFtpXCtlNotify Disconnected event.

Disconnected Event

See Also Frequently Asked Questions

Description

Fires after disconnecting from the FTP server.

Syntax

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

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

Part	Description
object	An FTP/X control.
index	An integer that identifies a control if it's in a control array.

Remarks

Fires when the State property changes to *StateNotConnected* (0). Note -- the Done event also fires *after* the Disconnected event.

IFtpXObjNotify Interface Disconnected Event

See Also Frequently Asked Questions

Description

Fires after disconnecting from the FTP server.

Syntax

Sub *object*_**Disconnected(**[*index* **As Integer**,] *FtpXObject* **As IFtpXObj)**

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

Part	Description
object	Required. An instance of an IFtpXObjNotify interface.
FtpXObject	A Ftp/X COM Object.

Remarks

Fires when the State property changes to *StateNotConnected* (0). Note -- the IFtpXObjNotify Done event also fires *after* the IFtpXObjNotify Disconnected event.

Disconnect Method

See Also Frequently Asked Questions

Description

Disconnects (logs off) from the server.

Syntax

object.**Disconnect**

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

Part	Description
object	Required. An FTP/X control.

Remarks

When connected to a server, the Disconnect method simply terminates the logical TCP/IP connection with the server. Upon disconnection, the control fires the Disconnected event.

RFC equivalent - QUIT

DisablePasv Property

See Also Frequently Asked Questions

Description

Used to enable or disable passive connections.

Syntax

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

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

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

Remarks

The FTP protocol uses two sockets when performing transfers -- one for sending commands (called the control socket) and one for transferring the data (communication socket). When the control socket issues a command that requires a communication socket, the FTP client and server need to negotiate a port to use.

The PORT command specifies that the *client* will determine which port to use for the communication socket. This is called an *active* connection.

The PASV command specifies that the *server* will determine which port to use for the communication. This is called a *passive* connection.

The DisablePASV property determines whether to use passive or active connections. When DisablePASV = False (default), passive connections are attempted. When DisablePASV = True, active connections are attempted. The DisablePASV property should be set to the desired value before invoking the Connect method and should not be changed while connected to the server.

Although the FTP RFC allows both modes, some servers or firewalls may only allow passive connections while others only allow active connections. Since there is no way to know ahead of time, you should first try one method and if it fails or times out then you will need to Disconnect and re-connect using the alternate mode.

Data Type

Boolean

Done Event

See Also Frequently Asked Questions

Description

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

Syntax

Sub *object*_**Done(**[*index* **As Integer**,] *LastMethod* **As MethodsEnum**, *ErrorCode* **As Integer**, *ErrorString* **As String)**

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

Part	Description
object	An FTP/X control.
index	An integer that identifies a control if it's in a control array.
LastMethod	An integer indicating the method which caused the event to fire.
ErrorCode	An integer that indicates whether the method completed successfully or not.
ErrorString	A string expression that gives a description of the error (if any) for the method that just completed.

Remarks

This event fires after each method completes. The *LastMethod* parameter indicates which method caused the event to fire. If the ErrorNumber parameter is zero, the operation completed correctly. If ErrorNumber is not zero, then Error specifies an error code.

The LastMethod parameter may be one of the following values:

Constant	Value	Description
FtpActionAbort	1	Abort Method.
FtpActionAllocate	2	Allocate Method.
FtpActionAppend	3	Append Method.
FtpActionChangeDir	4	ChangeDir Method.
FtpActionConnect	5	Connect Method.
FtpActionCreateDir	6	CreateDir Method.
FtpActionDelete	7	Delete Method.
FtpActionDeleteDir	8	DeleteDir Method.
FtpActionDisconnect	9	Disconnect Method.
FtpActionGetCurrentDir	10	GetCurrentDir Method.
FtpActionGetDirList	11	GetDirList Method.
FtpActionGetFile	12	GetFile Method.
FtpActionGetFilenameList	13	GetFilenameList Method.
FtpActionNone	0	No action.
FtpActionParentDir	14	ParentDir Method.
FtpActionPutFile	15	PutFile Method.
FtpActionPutUniqueFile	16	PutUniqueFile Method.
FtpActionQuote	22	Quote Method.
FtpActionReinitialize	20	Reinitialize Method.
FtpActionRename	19	Rename Method.
FtpActionSendType	17	SendType Method.

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.

IFtpXCtlNotify Interface Done Event

See Also Frequently Asked Questions

Description

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

Syntax

Sub *object*_**Done(**[*index* **As Integer**,] *FtpXControl* **As ByRef IFtpXCtl**, *LastMethod* **As MethodsEnum**, *ErrorCode* **As Integer**, *ErrorString* **As String)**

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

Part	Description
object	Required. An instance of an IFtpXCtlNotify interface.
FtpXControl	A Ftp/X control
LastMethod	An integer indicating the method which caused the event to fire.
ErrorCode	An integer that indicates whether the method completed successfully or not.
ErrorString	A string expression that gives a user-friendly version of the error (if any) for the method that just completed.

Remarks

This event fires after each method completes. The *LastMethod* parameter indicates which method caused the event to fire. If the ErrorNumber parameter is zero, the operation completed correctly. If ErrorNumber is not zero, then Error specifies an error code.

The LastMethod parameter may be one of the following values:

Constant	Value	Description
FtpActionAbort	1	Abort Method.
FtpActionAllocate	2	Allocate Method.
FtpActionAppend	3	Append Method.
FtpActionChangeDir	4	ChangeDir Method.
FtpActionConnect	5	Connect Method.
FtpActionCreateDir	6	CreateDir Method.
FtpActionDelete	7	Delete Method.
FtpActionDeleteDir	8	DeleteDir Method.
FtpActionDisconnect	9	Disconnect Method.
FtpActionGetCurrentDir	10	GetCurrentDir Method.
FtpActionGetDirList	11	GetDirList Method.
FtpActionGetFile	12	GetFile Method.
FtpActionGetFilenameList	13	GetFilenameList Method.
FtpActionNone	0	No action.
FtpActionParentDir	14	ParentDir Method.
FtpActionPutFile	15	PutFile Method.
FtpActionPutUniqueFile	16	PutUniqueFile Method.
FtpActionQuote	22	Quote Method.
FtpActionReinitialize	20	Reinitialize Method.
FtpActionRename	19	Rename Method.
FtpActionSendType	17	SendType Method.

IFtpXObjNotify Interface Done Event

See Also Frequently Asked Questions

Description

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

Syntax

Sub *object*_**Done(**[*index* **As Integer**,] *FtpXObject* **As IFtpXObj**, *LastMethod* **As MethodsEnum**, *ErrorCode* **As Integer**, *ErrorString* **As String)**

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

Part	Description
object	Required. An instance of an IFtpXObjNotify interface.
FtpXObject	A Ftp/X COM Object.
LastMethod	An integer indicating the method which caused the event to fire.
ErrorCode	An integer that indicates whether the method completed successfully or not.
ErrorString	A string expression that gives a user-friendly version of the error (if any) for the method that just completed.

Remarks

This event fires after each method completes. The *LastMethod* parameter indicates which method caused the event to fire. If the ErrorNumber parameter is zero, the operation completed correctly. If ErrorNumber is not zero, then Error specifies an error code.

The LastMethod parameter may be one of the following values:

Constant	Value	Description
FtpActionAbort	1	Abort Method.
FtpActionAllocate	2	Allocate Method.
FtpActionAppend	3	Append Method.
FtpActionChangeDir	4	ChangeDir Method.
FtpActionConnect	5	Connect Method.
FtpActionCreateDir	6	CreateDir Method.
FtpActionDelete	7	Delete Method.
FtpActionDeleteDir	8	DeleteDir Method.
FtpActionDisconnect	9	Disconnect Method.
FtpActionGetCurrentDir	10	GetCurrentDir Method.
FtpActionGetDirList	11	GetDirList Method.
FtpActionGetFile	12	GetFile Method.
FtpActionGetFilenameList	13	GetFilenameList Method.
FtpActionNone	0	No action.
FtpActionParentDir	14	ParentDir Method.
FtpActionPutFile	15	PutFile Method.
FtpActionPutUniqueFile	16	PutUniqueFile Method.
FtpActionQuote	22	Quote Method.
FtpActionReinitialize	20	Reinitialize Method.
FtpActionRename	19	Rename Method.
FtpActionSendType	17	SendType Method.

DstFilename Property

See Also Frequently Asked Questions

Description

Filename for destination of transfers.

Syntax

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

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

Part	Description
object	An FTP/X control.
DstFilename	A string expression containing the filename and path for file related methods.

Remarks

The GetFile, PutFile, Append, Rename and Delete methods all require a destination filename. The DstFilename property may contain both the filename and a path.

When performing a GetFile, the DstFilename is the name of the file on the local machine. For example:

```
FtpX.DstFilename = "C:\temp\somefile.txt" 'the local file
FtpX.SrcFilename = "somefile.txt" 'the server file
FtpX.GetFile
```

When performing a PutFile or Append, the DstFilename is the name of the file on the remote machine (the FTP server). For example:

```
FtpX.SrcFilename = "C:\temp\somefile.txt" 'the local file
FtpX.DstFilename = "somefile.txt" 'the server file
FtpX.PutFile 'or FtpX.Append
```

When performing a Rename, the DstFilename is the name that the file will become on the remote machine (the new name on the FTP server). For example:

```
FtpX.SrcFilename = "oldfile.txt" 'the original server filename
FtpX.DstFilename = "newfile.txt" 'the new server filename
FtpX.Rename
```

Data Type

String

GetCurrentDir Method

See Also Frequently Asked Questions

Description

Retrieves the ftp server's current working directory.

Syntax

object.**GetCurrentDir**

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

Part	Description
object	Required. An FTP/X control.

Remarks

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

RFC equivalent - PWD

GetDirList Method

See Also Frequently Asked Questions

Description

Gets a directory listing from the server.

Syntax

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

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

Part	Description
object	Required. An FTP/X 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. Use the *pattern* parameter to limit the number of responses. If the *pattern* parameter is not specified, then the Pattern property is used. If both the *pattern* parameter and the Pattern property are empty, all directory items will be returned.

RFC equivalent - LIST

GetFile Method

See Also Frequently Asked Questions

Description

Get 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/X 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.

Both the *srcfilename* and *dstfilename* parameters can include paths. If the *srcfilename* and *dstfilename* parameters are not specified, the SrcFilename property and DstFilename property values are used.

RFC equivalent - RETR

GetFilenameList Method

See Also Frequently Asked Questions

Description

Gets a filename list from the server.

Syntax

object.**GetFilenameList***Pattern*

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

Part	Description
object	Required. An FTP/X control.
Pattern	Optional. A string expression containing a pattern to use when retrieving the list of filenames.

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. Use the *pattern* parameter to limit the number of responses. If the *pattern* parameter is not specified, then the Pattern property is used. If both the *pattern* parameter and the Pattern property are empty, all filenames will be returned.

RFC equivalent - NLST

Host Property

See Also Frequently Asked Questions

Description

Name or IP address of the FTP server.

Syntax

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

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

Part	Description
object	An FTP/X control.
Host	A string expression that specifies the name or IP address of a server.

Remarks

Used to specify the name (or IP address directly) of the server to use during execution of the next Connect method. If using a name, the fully qualified domain name (such as "ftp.mabry.com") is recommended.

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

IFtpXCtlNotify Interface

See Also Events Frequently Asked Questions

Description

Fast Notification interface for the OCX control.

Remarks

The FTP/X OCX implements a Fast Notification (early-bound callback) interface as an alternative to using events. You can implement the early-bound callback functions which are identical to the events except that an instance of the OCX calling the function is passed as the first parameter. This extra parameter substitutes for the Index parameter you would have had in a control array implementation.

To implement the IFtpXCtlNotify interface, add the following in the declarations section of a form or class:

```
Implements IFtpXCtlNotify
```

Then, use the NotificationObject property to specify the object which will receive the notifications, such as:

```
Private Sub Form_Load ()
    FtpXCtl1.NotificationObject = Me
End Sub
```

Finally, use the various events such as IFtpXCtlNotify Done event, the IFtpXCtlNotify StateChanged event, etc. to capture the events.

IFtpXObjNotify Interface

See Also Events Frequently Asked Questions

Description

Fast Notification interface for the Ftp/X COM Object (DLL).

Remarks

The FTP/X COM Object implements a Fast Notification (early-bound callback) interface as an alternative to using events. You can implement the early-bound callback functions which are identical to the events except that an instance of the object calling the function is passed as the first parameter. This extra parameter substitutes for the Index parameter you would have had in an array implementation.

To implement the IFtpXObjNotify interface, add the following in the declarations section of a form or class:

```
Dim WithEvents FtpX As FtpXObj

Implements IFtpXObjNotify
```

Then, use the NotificationObject property to specify the object which will receive the notifications, such as:

```
Private Sub Form_Load ()
    Set FtpX = New FtpXObj
    FtpX.NotificationObject = Me
End Sub
```

Finally, use the various events such as IFtpXObjNotify Done event, the IFtpXObjNotify StateChanged event, etc. to capture the events.

LastError Property

See Also Frequently Asked Questions

Description

Holds the last error number reported.

Syntax

object.**LastError** [= *LastError*]

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

Part	Description
object	An FTP/X control.
LastError	An integer containing the error code from the last method executed.

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

LastErrorString Property

See Also Frequently Asked Questions

Description

Holds a description of the last error number reported.

Syntax

object.**LastErrorString** [= *LastErrorString*]

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

Part	Description
object	An FTP/X control.
LastErrorString	A string expression that gives a description of the error (if any) for the method that just completed.

Remarks

This property contains a user-friendly version of the result of the last method executed. It is empty if the last method completed without error. Otherwise, it contains a description of the error code in the LastError property.

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

LastMethod Property

See Also Frequently Asked Questions

Description

Returns the last method executed.

Syntax

object.**LastMethod** [= *LastMethod*]

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

Part	Description
object	An FTP/X control.
LastMethod	An integer which represents the last method executed.

Remarks

This property returns an integer which represents the last method executed.

The LastMethod property may be one of the following values:

Constant	Value	Description
FtpActionAbort	1	Abort Method.
FtpActionAllocate	2	Allocate Method.
FtpActionAppend	3	Append Method.
FtpActionChangeDir	4	ChangeDir Method.
FtpActionConnect	5	Connect Method.
FtpActionCreateDir	6	CreateDir Method.
FtpActionDelete	7	Delete Method.
FtpActionDeleteDir	8	DeleteDir Method.
FtpActionDisconnect	9	Disconnect Method.
FtpActionGetCurrentDir	10	GetCurrentDir Method.
FtpActionGetDirList	11	GetDirList Method.
FtpActionGetFile	12	GetFile Method.
FtpActionGetFilenameList	13	GetFilenameList Method.
FtpActionNone	0	No action.
FtpActionParentDir	14	ParentDir Method.
FtpActionPutFile	15	PutFile Method.
FtpActionPutUniqueFile	16	PutUniqueFile Method.
FtpActionQuote	22	Quote Method.
FtpActionReinitialize	20	Reinitialize Method.
FtpActionRename	19	Rename Method.
FtpActionSendType	17	SendType Method.

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

Data Type

Integer

LibraryName Property

See Also Frequently Asked Questions

Description

Specifies the name of the socket stack (winsock) DLL.

Syntax

object.**LibraryName** [= *LibraryName*]

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

Part	Description
object	An FTP/X control.
LibraryName	A string expression containing the winsock library to use.

Remarks

The default value for this property is "wsock32.dll", which is the socket stack that Microsoft ships with 32-bit systems. If you have a different socket stack provider, you can use the provider's socket stack by specifying the provider's DLL name.

Data Type

String

LogonName Property

See Also Frequently Asked Questions

Description

Username for logon process.

Syntax

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

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

Part	Description
object	An FTP/X control.
LogonName	A string expression containing the user's logon name.

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 Frequently Asked Questions

Description

Password for server logon process.

Syntax

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

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

Part	Description
object	An FTP/X control.
LogonPassword	A string expression containing the user's password.

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

NotificationObject Property

See Also Frequently Asked Questions

Description

Specifies the object that has implemented an interface to receive the Fast Notifications.

Syntax

object.**NotificationObject** [= *NotificationObject*]

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

Part	Description
object	An FTP/X control.
NotificationObject	Object that will receive the Fast Notifications.

Remarks

Both the FTP/X OCX and COM object support fast-notification callback interfaces. If you implement IFtpXCtlNotify or IFtpXObjNotify in a form or class, you can assign an instance of that form or class to the FTP/X object or control's NotificationObject. This will allow the form or class to receive event notifications directly.

For example, to implement Fast Notifications for the FTP/X control for a form, implement the interface in the Declarations section with:

```
    Implements IFtpXCtlNotify
```

Then, to have the form receive the notifications, do:

```
Private Sub Form_Load ()
    FtpXCtl1.NotificationObject = Me
End Sub
```

Data Type

IFtpXCtlNotify

ParentDir Method

See Also Frequently Asked Questions

Description

Changes the server's working directory from the current directory to its parent directory.

Syntax

object.**ParentDir**

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

Part	Description
object	Required. An FTP/X control.

Remarks

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

RFC equivalent - CDUP

Pattern Property

See Also Frequently Asked Questions

Description

Pattern for directory and filename listings.

Syntax

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

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

Part	Description
object	An FTP/X control.
Pattern	A string expression containing a pattern for limiting directory of file listings.

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 Frequently Asked Questions

Description

Determines to which port the control connects.

Syntax

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

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

Part	Description
object	An FTP/X control.
Port	An integer that specifies the port number to use to connect to the 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

IFtpXCtlNotify Interface Progress Event

See Also Frequently Asked Questions

Description

Fires as lengthy operations progress.

Syntax

Sub *object*_**Progress(**[*index* **As Integer**,] *FtpXControl* **As ByRef IFtpXCtl**, *BytesTransfered* **As Long)**

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

Part	Description
object	Required. An instance of an IFtpXCtlNotify interface.
FtpXControl	A Ftp/X control
BytesTransfered	A long integer indicating the number of bytes transferred thus far.

Remarks

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

IFtpXObjNotify Interface Progress Event

See Also Frequently Asked Questions

Description

Fires as lengthy operations progress.

Syntax

Sub *object*_**Progress(**[*index* **As Integer**,] *FtpXObject* **As IFtpXObj**, *BytesTransfered* **As Long)**

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

Part	Description
object	Required. An instance of an IFtpXObjNotify interface.
FtpXObject	A Ftp/X COM Object.
BytesTransfered	A long integer indicating the number of bytes transferred thus far.

Remarks

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

Progress Event

See Also Frequently Asked Questions

Description

Fires as lengthy operations progress.

Syntax

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

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

Part	Description
object	An FTP/X control.
index	An integer that identifies a control if it's in a control array.
BytesTransfered	A long integer indicating the number of bytes transferred thus far.

Remarks

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

PutFile Method

See Also Frequently Asked Questions

Description

Copies 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/X control.
SrcFilename	Optional. A string expression containing the name and path of the file on the local machine.
DstFilename	Optional. A string expression containing the name of the file on the server.

Remarks

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

Both the *srcfilename* and *dstfilename* parameters can include paths. If the *srcfilename* and *dstfilename* parameters are not specified, the SrcFilename property and DstFilename property values are used.

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

RFC equivalent - STOR

PutUniqueFile Method

See Also Frequently Asked Questions

Description

Copies a file to the FTP server and the server creates unique filename.

Syntax

object.**PutUniqueFile***SrcFilename*

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

Part	Description
object	Required. An FTP/X control.
SrcFilename	Optional. A string expression containing the name and path of the file on the local machine.

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. The server will return the name of the file to be created as part of its response to the PutUniqueMethod. Use the Debug event to capture the response if desired.

If the *srcfilename* parameter is not specified, the SrcFilename property is used.

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

RFC equivalent - STOU

Quote Method

See Also 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/X 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 raw command string to the server. This method is useful for sending extra or non-standard commands, such as:

```
FtpX.Quote "HELP" & vbCrLf
```

Or,

```
FtpX.QuoteCmd = "SITE chmod 666 /somedir/filename.txt" & vbCrLf
FtpX.Quote
```

If the *Command* parameter is omitted, the control uses the data in the QuoteCmd property. After the server finishes responding, the Done event is fired. Upon completion of the Quote command, the ASCII response returned by the server can be read from the ReadData property.

QuoteCmd Property

See Also Frequently Asked Questions

Description

Command sent to the server with the Quote method.

Syntax

object.**QuoteCmd** [= *QuoteCmd*, *command*]

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

Part	Description
object	An FTP/X control.
QuoteCmd	A string expression that specifies the command to send to the server.
command	A string expression that specifies the command to send to the FTP server.

Remarks

You can use this control's Quote method to send arbitrary commands to the server. This is often used to invoke additional or non-standard commands, such as:

```
FtpX.QuoteCmd = "HELP" & vbCrLf
FtpX.Quote
```

Or,

```
FtpX.QuoteCmd = "SITE chmod 666 /somedir/filename.txt" & vbCrLf
FtpX.Quote
```

Data Type

String

ReadData Property

See Also Frequently Asked Questions

Description

Server response from the Quote method.

Syntax

object.**ReadData**

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

Part	Description
object	An FTP/X 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 Quote method lets you directly send a command to the server and retrieve its response. Upon successful completion of the Quote method the ReadData property will contain the server's response.

Data Type

String

Recordset Property

See Also Frequently Asked Questions

Description

An ADO recordset of the items returned by the GetDirList method.

Syntax

object.**Recordset** [= *Recordset*]

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

Part	Description
object	An FTP/X control.
Recordset	A Recordset of the items returned by a GetDirList or GetFilenameList.

Remarks

The Recordset property is an ADO recordset of the items returned by the GetDirList method. After executing the method you can assign the Recordset property like any other recordset.

The DirItemPattern property is used to specify how the server's response is parsed into the recordset's fields. The default value of DirItemPattern should be sufficient for most Unix and NT-based servers.

Data Type

Recordset

Reinitialize Method

See Also Frequently Asked Questions

Description

Terminate and re-establish ftp session.

Syntax

object.**Reinitialize***LogonName*,*LogonPassword*,*Account*

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

Part	Description
object	Required. An FTP/X control.
LogonName	Optional. A string expression containing the logon name to use.
LogonPassword	Optional. A string expression containing the password to use.
Account	Optional. A string expression containing the account name to use.

Remarks

Terminates the current user's FTP session and starts a new session. If the *logonname, logonpassword, and account* parameters are not specified the LogonName, LogonPassword, and Account properties are used. See the Connect.

RFC equivalent - REIN

Rename Method

See Also Frequently Asked Questions

Description

Renames a file on remote system.

Syntax

object.**Rename***SrcFilename*,*DstFilename*

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

Part	Description
object	Required. An FTP/X control.
SrcFilename	Optional. A string expression containing the original server filename.
DstFilename	Required. A string expression containing the new server filename.

Remarks

Renames the remote (server) file named by the *srcfilename* parameter to the name contained in the *dstfilename* parameter. If *srcfilename* and *dstfilename* are not specified, the SrcFilename and DstFilename properties are used.

RFC equivalent - RNFR and RNTO

SendType Method

See Also Frequently Asked Questions

Description

Sends transfer type setting to server.

Syntax

object.**SendType***Type*

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

Part	Description
object	Required. An FTP/X control.
Type	Required. Integer specifying 0 (ASCII) or 1 (Binary)

Remarks

Sets the server's transfer mode to either ASCII or Binary. If the *Type* parameter is set to 0 or FtpTypeAscii, the server is set to transfer in ASCII mode. If *Type* is set to 1 or FtpTypeBinary, the server is set to transfer in Binary mode. If *Type* is omitted, the value of the Type property is used.

RFC equivalent - TYPE

SrcFilename Property

See Also Frequently Asked Questions

Description

Source for transfer oriented methods.

Syntax

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

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

Part	Description
object	An FTP/X control.
SrcFilename	A string expression containing a filename and path for file related methods.

Remarks

The GetFile, PutFile, PutUniqueFile, Append, and Rename methods all require a source filename. The SrcFilename property may contain both the filename and a path.

When performing a PutFile (or Append or PutUniqueFile), the SrcFilename is the name of the file on the local machine. For example:

```
FtpX.SrcFilename = "C:\temp\somefile.txt" 'the local file
FtpX.DstFilename = "somefile.txt" 'the server file
FtpX.PutFile
```

When performing a GetFile, the SrcFilename is the name of the file on the remote machine (the FTP server). For example:

```
FtpX.DstFilename = "C:\temp\somefile.txt" 'the local file
FtpX.SrcFilename = "somefile.txt" 'the server file
FtpX.GetFile
```

When performing a Rename, the SrcFilename is the server's original filename that will change (the old name on the FTP server). For example:

```
FtpX.SrcFilename = "oldfile.txt" 'the original server filename
FtpX.DstFilename = "newfile.txt" 'the new server filename
FtpX.Rename
```

Data Type

String

State Property

See Also Frequently Asked Questions

Description

Returns the current state of the control.

Syntax

object.**State** [= *State*]

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

Part	Description
object	An FTP/X control.
State	Integer that represents the current state of the 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.

Remarks

This property may take on many values while executing commands.

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

The following constants represent the set of values that can be assigned to the State property:

Constant	Value	Description
StateNoConnected	0	Not currently connected to the server.
StateConnected	1	Connected to the server.
StateConnecting	2	Connecting to the server.
StateSentUser	3	Logging on -- username sent.
StateSentPassword	4	Logging on -- password sent.
StateSentAccount	5	Logging on -- account sent.
StateDisconnecting	6	Disconnecting from the server.
StateReading	7	Reading data from the server.
StateSendData	8	Sending data to the server.
StateAllocating	9	Allocating space on the server.
StateCreatingDir	10	Creating a directory on the server.
StateChangingDir	11	Changing the server's working directory.
StateDeleting	12	Deleting a file from the server.
StateDeletingDir	13	Deleting a directory from the server.
StateGettingCurrentDir	14	Retrieving the current directory name.
StateGettingDataSocket	15	Negotiating the data socket with server.
StateGotDataSocket	16	Have negotiated the data socket with server.
StateGettingDirList	17	Getting listing.
StateGettingFile	18	Getting a file.
StatePuttingFile	19	Putting a file.
StateRenameFromSent	20	Renaming file -- specifying file to rename.
StateRenameToSent	21	Renaming file -- specifying file's new name.
StateAborting	22	Aborting a transfer.

StateSentType	23	Setting the server's transfer mode.
StateExecQuote	24	Sending a raw command.

Data Type

Integer

StateChanged Event

See Also Frequently Asked Questions

Description

This event procedure is fired as the control executes its various methods.

Syntax

Sub *object*_**StateChanged(**[*index* **As Integer**,] *NewState* **As StatesEnum**, *OldState* **As StatesEnum)**

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

Part	Description
object	An FTP/X control.
index	An integer that identifies a control if it's in a control array.
NewState	Integer representing the current state of the control.
OldState	Integer representing the previous state of the control.

Remarks

As various methods execute, the StateChanged event fires to indicate the current state of the control.

The *OldState* and *NewState* parameters contain the previous and current states. See the State property for a list of the possible values.

IFtpXObjNotify Interface StateChanged Event

See Also Frequently Asked Questions

Description

This event procedure is fired as the control executes its various methods.

Syntax

Sub *object*_**StateChanged(**[*index* **As Integer**,] *FtpXObject* **As IFtpXObj**, *NewState* **As StatesEnum**, *OldState* **As StatesEnum)**

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

Part	Description
object	Required. An instance of an IFtpXObjNotify interface.
FtpXObject	A Ftp/X COM Object.
NewState	Integer representing the current state of the control.
OldState	Integer representing the previous state of the control.

Remarks

As various methods execute, the StateChanged event fires to indicate the current state of the control.

The *OldState* and *NewState* parameters contain the previous and current states. See the State property for a list of the possible values.

IFtpXCtlNotify Interface StateChanged Event

See Also Frequently Asked Questions

Description

This event procedure is fired as the control executes its various methods.

Syntax

Sub *object*_**StateChanged(**[*index* **As Integer**,] *FtpXControl* **As ByRef IFtpXCtl**, *NewState* **As StatesEnum**, *OldState* **As StatesEnum)**

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

Part	Description
object	Required. An instance of an IFtpXCtlNotify interface.
FtpXControl	A Ftp/X control.
NewState	Integer representing the current state of the control.
OldState	Integer representing the previous state of the control.

Remarks

As various methods execute, the StateChanged event fires to indicate the current state of the control.

The *OldState* and *NewState* parameters contain the previous and current states. See the State property for a list of the possible values.

StateString Property

See Also Frequently Asked Questions

Description

A user-friendly version of the current state of the control.

Syntax

object.**StateString** [= *StateString*]

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

Part	Description
object	An FTP/X control.
StateString	A string expression containing a descriptive state of the control.

Remarks

This property returns a string expression that tells the current state of the control. See the State property for a list of the State values.

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

Data Type

String

Timeout Property

See Also Frequently Asked Questions

Description

The amount of time (in seconds) the control will wait for an operation..

Syntax

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

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

Part	Description
object	An FTP/X control.
Timeout	A long integer that specifies the amount of time to wait, in seconds.

Remarks

This property determines how long the control will wait for various methods to finish (before declaring an error). This property is set in seconds.

Data Type

Long

Type Property

See Also Frequently Asked Questions

Description

Specifies the transfer type - binary or ascii.

Syntax

object.**Type** [= *enumtype*]

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

Part	Description
object	An FTP/X control.
enumtype	Integer representing the transfer type.

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

enumType

Version Property

See Also Frequently Asked Questions

Description

Returns the version of the control.

Syntax

object.**Version** [= *Version*]

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

Part	Description
object	An FTP/X control.
Version	TODO

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

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.
503 316th Street Northwest
Stanwood, WA 98292

Phone: 360-629-9278
Fax: 360-629-9278

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.