

WSNWUTIL Help Contents



Software Solutions

[Welcome](#)

[Location of latest WSNWUTIL.ZIP release](#)

[To users of the WSNWDEMO application](#)

[Configuration](#)

[WSN_CONN - Connectivity Testing](#)

[WSN_ECHO - An ECHO Server](#)

[WSN_FTPC - File Transfer Protocol Client](#)

[WSN_FNGR - Finger Client](#)

[WSN_FNGS - A Finger Server](#)

[What is this WSNETWRK that keeps being mentioned?](#)

For Help on Help, Press F1

Glossary

Welcome



Welcome! Thanks for trying out the WSNETWRK Utility package - **WSNWUTIL**.

This is the second go-round for these applications. The first was named WSNWDEMO.ZIP and was distributed around via the internet. That version had two (or more) basic problems;

1. It was a series of MDI child windows - which in itself is not a problem, unless of course you are me and you have to maintain it. It was just getting too big. Version 3 may be back to the MDI format as I am very pleased with the newest Visual C++ and MFC 3.0, and how easy it is to manage such a project.
2. It was based on the initial, Alpha release of my WSNETWRK SDK (more about that later). Over the summer of 94 WSNETWRK itself was overhauled in preparation for a commercial release. This version of the WSNWUTIL tools all use the new (and improved) foundation code and is the better for it.
- ?. (There would be an infinity character here if I could find one) WSNWUTIL had little in the way of error checking and feedback, and definitely was lacking something in the visual way. There were also a few bugs (bugs??!? Me???) in the underlying WSNETWRK code. As time goes on, I test the WSNETWRK code on more and more TCP/IP stacks. If these applications do not work on your system, it is likely that I have not had access to your TCP/IP vendors stack for testing. Definitely drop me a line if you have problems. The only way I can know what to fix is if someone tells me.

Anyway, thanks again. WSNWUTIL will grow over time, with new clients and servers being added as I expand the WSNETWRK Internet Protocol API libraries.

These applications and the underlying DLLs are freely distributable in original form. No fees may be charged for distribution or use without the express written consent of DFM. If you have questions about including WSNWUTIL in a collection of internet utilities or as part of a commercial product, please contact us at mclouden@dfm.com.

I have explained this before but it bears repeating - WSNWUTIL is a **TEST** application. I wrote it and will continue to expand it so that I may test out the APIs that I write. The apps are not perfect in themselves, but I have tried to clean them up as best I can/have time for. I am not directly in the end-user application business. My specialty is middle-ware. The upside of WSNWUTIL being a test application is that as I expand the IP API selection (soon to include Gopher, NNTP, and SMTP/POP3), there will be new additions to the WSNWUTIL collection.

A note to any who may have previously installed WSNWDEMO; Remove the WSNETWRK.DLL and related DLLs from your path. The most recent version of WSNETWRK is named **WSNET100.DLL**. The original WSNETWRK.DLL has been retired - it was after all an alpha release.

Please feel free to mail me with any comments you may have, but most importantly - send in any bug reports. I have done all I can to find them on my own, but I need your help.

I can be reached at mclouden@dfm.com.

Mark Clouden, DFM
November 3, 1994

What is this WSNETWRK that keeps being mentioned?

WSNETWRK is a Microsoft Windows Dynamic Link Library (DLL) that encapsulates and expands upon the functionality of the Windows Sockets (Winsock) API.

My main goal, my primary focus - is on the development of a series of internet APIs. These allow a developer to embed internet functionality into a program with minimum effort. The best example is Word for Windows 6.0 - on the File menu is an option to send the document as mail.

The key is to make it easy since you will want to keep your focus on whatever killer-app you are developing and **not** on how to transfer a file, or send mail. That is my job.

Besides the obvious benefit of not having to know how the File Transfer Protocol works, WSNETWRK also works over most Winsocks. Even though there is a specification for how to implement a Winsock interface, there are always variances, some more minor than others. So you can write a single code source that will run over any vendors network software.

WSNETWRK allows for all the functionality of raw Winsock calls, it imposes no restrictions on what you can and cannot do with a socket.

All I/O operations use the Winsock non-blocking mechanism. If you have ever written an application using this method, you know that it is not a pleasant experience. WSNETWRK completely changes the way you interact with a socket. Here is a simple example:

Using Winsock calls:

To receive data on a socket, you call `recv()`. If there is data in the buffer waiting to be read, it returns that data to you and indicates its success by returning the number of bytes just read. Simple huh?

Now, what if there is no data? More often than not, there isn't any and `recv()` returns an error indicator. You examine the error code and handle recovery if it is in fact an error. But, it may be a code that tells you that there is no data - this is not an error, just a warning. Now, to know when data is available, you will call `WSAAsyncSelect()` and specify that you want to be alerted when data is available. Great.

In your window (which you must have associated with the socket), you handle the event message that is telling you that data is available. You call `recv()` again, and again check for errors, but most likely it completed OK and the data is in the buffer you specified. Absolutely fabulous.

So you must handle error recovery in two places, the initial `recv()` and the subsequent one. You must also be prepared to process the received data in both places. And of course, you must have a window to process the event messages generated by Winsock.

Now I won't try to mislead you into thinking this is rocket science. It's not. But it is just plain ugly. It retains almost none of the event-driven nature of Windows, and a complicated protocol just makes things worse. On top of this, the Winsock specification does not allow callback functions.

Of course, you can just use blocking sockets. Call `recv()` and your program waits on the data. Of course, nothing is free, and there are drawbacks to this method also.

Using WSNETWRK:

So you want to receive some data? Call the `WSRecv()` (or `WSRecvFrom()`) function. WSNETWRK does everything described above for you.

When the data has been received (or has failed), you will receive **either** a message sent

to a window, or a call to the function of your choosing.

Thats it.

Of course, having the opportunity (I am not bound by any specifications), I have added logical operations (receive a line of text delimited by carriage return/linefeed, or receive a block of text delimited as in the NNTP, and SMTP protocols; delimited by a line of text consisting of only a period). How about receives with time-out values?

There is more to it than just changing the interface to the socket; What about allocating memory associated with a socket in the same way that memory is associated with a window? You can access the memory using calls similar to the Windows API GetWindowLong()/SetWindowLong(). This is a great place for I/O buffers. Since timers are at a premium in non-NT versions of Windows, there is a virtual timer interface built in.

OK. Enough. If you are interested, fire off some Email to mclouden@dfm.com and let me know. Ill send you the full information packet. Also, WSNETWRK is presented in the Dr. Dobbs SourceBook for Winter 1994 - Information Highway (page 50).

You may want to know right off the bat that WSNETWRK is not freeware or shareware, it is a commercial product, reasonably priced and well worth it.

Four reasons to at least *think* about getting more information about WSNETWRK;

1. WSNETWRK handles the various inconsistencies that are found between a vendors Winsock and the Winsock specification. We guarantee that WSNETWRK will work over your TCP/IP stack (as long as its at least close to compliant) and if WSNETWRK does not work, we will fix it so that it does.
2. The performance of non-blocking sockets without the excruciating (OK, mildly excruciating) pain. On the subject of performance, the WSN_FTPC utility included in this collection runs just as fast as any other FTP client I have used. SO? I hear you say. Well, WSN_FTPC makes use of the WSNWFTPC API. That in turn makes use of the WSNETWRK API. And that of course makes all the native Winsock calls. There is a LOT of work going on behind the scenes. And the performance hit for encapsulating all that functionality is almost nil. A rough benchmark is that WSN_FTPC can transfer 400K per second on an ethernet LAN. Since each send is 512 bytes, that comes to 800 sends or receives per second. Yow! (Well, I was impressed).
3. When Winsock 2.0 is unveiled, it will expand the functionality available to you if you use Winsock. Well, I read the news to, and the code to support shared sockets, and lots of the neat enhancements coming our way is already waiting to be enabled in WSNETWRK. And when those enhancements do come, you can bet that we will have a list of our own. Thats a benefit of not being bound by a specification. How long has Winsock 1.1 been king? How long before we actually see Winsock 2.0? Committees move slow. We can enhance our APIs as the need/demand arises.
4. Unlimited runtime distribution. You pay for each developer who will use the API. You do not pay one thin dime to distribute your finished application. I dont care if you (and encourage you to) write the next Mosaic, and sell a million copies. Thats your money, not mine. Distribution fees always struck me as a ridiculous concept. I want no part of that.

What else?

Ive also developed other APIs, built using the WSNETWRK module itself. Currently these include Connectivity testing (Echo, Ping), Finger (client and server), and FTP client support.

These modules are available now, and can be purchased at a discount when purchased with a WSNETWRK license, or seperately at prices I think are appropriate for the level of functionality

included in each module.

Slated for development over the next few months are Gopher client, NNTP client, SMTP/POP3 mail client and HTTP client modules.

OK, one last bit of info - Who is this guy?:

Hi! Im Mark Clouden, President of DFM Software Solutions.

Fact of life: Its difficult to trust a young company. Will they be around tomorrow? Will they support the product? Will they manage changes to their product so that it has little impact on the customer? DFM will. Yes, definitely. And yes, weve done well with that so far and intend to continue.

As the developer, the responsibilities fall my way. I have been in software development since 1982. I am currently employed in the Research and Development department of a Fortune 100 company. I have been developing Winsock applications for a few years now, pretty much full time. I myself use the WSNETWRK APIs in all my development, and will be doing so again in my current project; the communications class structure for a series of client-server applications that will initially be used by over 10,000 users, and will serve as the foundation upon which all future client-server work the company does.

Networking is my business. Its what I do best. Its what I enjoy most. WSNETWRK will be around and so will I.

To users of the WSNWDEMO application

WSNWDEMO.ZIP was distributed via the internet in May of 1994. Over the summer, the underlying DLLs were redesigned. You should make sure that the files distributed with WSNWDEMO.ZIP are removed from your system (or at least your path). Specifically, these are WSNETWRK.DLL, WSNWFNGR.DLL and WSNWECHO.DLL.

The WSNWDEMO files have been replaced by WSNET100.DLL, WSNWCONN.DLL, WSNWFNGR.DLL and WSNWFTPC.DLL.

The WSNWDEMO application will **not** run correctly with the new DLLs. If you wish to retain WSNWDEMO on your system, keep it in a directory of its own.

WSN_CONN - Connectivity Testing

WSN_CONN integrates the standard Ping utility which uses ICMP as the communications protocol, with an ECHO protocol client. The reason for this integration is that many Winsock implementation do not support the creation of ICMP sockets.

WSN_CONN consists of a single window which looks like this; For help about individual controls in this window, click on the control.

The screenshot shows a window titled "WSN_CONN" with a menu bar containing "Font" and "Help". The main area contains the following controls:

- Select Host To Test:** A text box containing "198.106.97.11" with a dropdown arrow button to its right.
- Test Interval: (Seconds):** A text box containing "3".
- Echo Port Number:** A text box containing "0".
- Testing Protocol:** A group box containing three radio buttons: ☒ ICMP, ☐ TCP, and ☐ UDP.
- Buttons:** "Close", "Help", "New Host...", "Delete Host", "Start", and "Reset Stats".
- Statistics:** A list of statistics with their current values:
 - Attempts: 0
 - Late Replies: 0
 - Error Count : 0
 - Average Time: 0.0
 - Maximum Time: 0.0
 - Minimum Time: 0.0
- Status Bar:** A label at the bottom of the window that reads "Inactive".

WSN_ECHO - An ECHO Server

WSN_ECHO is an ECHO protocol server. Since I included an ECHO client (in the WSN_CONN application), I figured I had better include an ECHO server as well.

If you are running Windows NT, check if you have the basic TCP/IP services running. If so, then NT is providing ECHO server support and WSN_ECHO is not needed (although it will still run OK).

The screenshot shows the WSN_Echo application window. The title bar is blue with the text "WSN_Echo". Below the title bar is a menu bar with "Font" and "Help". The main area is divided into several sections. At the top, there are two buttons: "TCP/Start" and "UDP/Start". To the right of these buttons are two input fields: "Port ID:" with the value "0" and "Max Connects:" with the value "0". To the right of these input fields are two buttons: "Close" and "Help". Below the "TCP/Start" and "UDP/Start" buttons is a "Statistics" section. It contains two rows of text: "TCP Count: 0" and "Connect Count: 0" on the first row, and "UDP Count: 0" on the second row. To the right of the "Statistics" section is a button labeled "Reset Counts". Below the "Statistics" section is a label "Active TCP Connections:" followed by the value "0". At the bottom of the window is a large empty rectangular area. At the very bottom of the window is a button labeled "Kill Selected Sessions".

Port ID:	Max Connects:
0	0

Statistics
TCP Count: 0
UDP Count: 0
Connect Count: 0

Active TCP Connections: 0

Kill Selected Sessions

WSN_FTPC - File Transfer Protocol Client

This is the first cut of WSN_FTPC. All of the basic functionality of a FTP client is here. Because of time constraints, some advanced functions were removed for lack of testing. Most of what is disabled revolves around queuing of file transfers. This will be available soon in an update to the program.

From the main screen, you can set up host configurations, consisting of a configuration name, a host system name, a user ID and password, and some optional parameters.

Any changes to the configuration are automatically saved.

On selecting the [Connect](#) button, the main screen is hidden and the connection progress dialog box is displayed.

When you have connected to the host system and successfully logged on, the connection progress dialog is removed and the main console window is displayed.

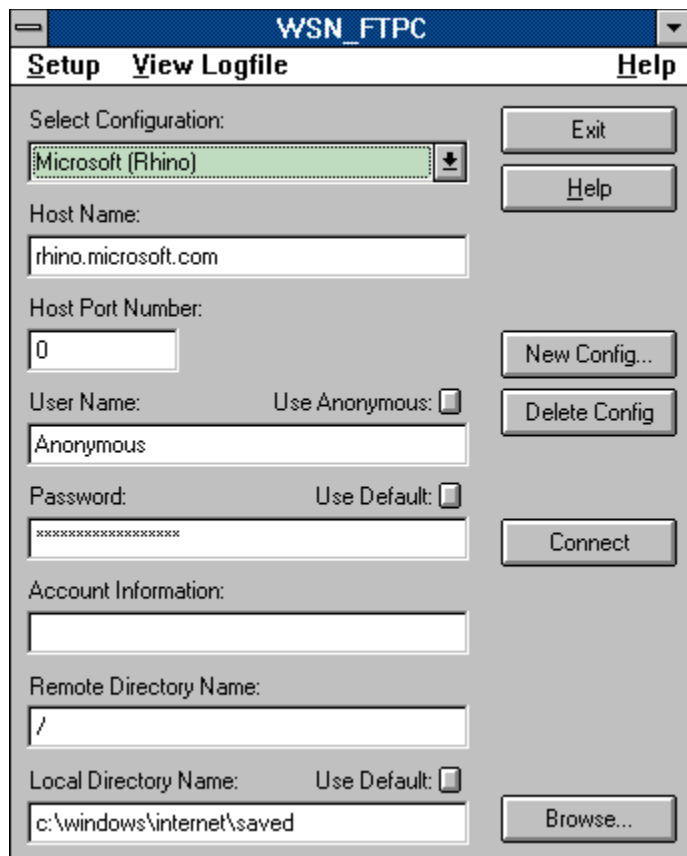
On disconnecting from the host, you are returned to the main window again.

Without the file queuing mechanism enabled, the console is limited in that only a single file may be selected for download at any one time. When the queuing function is enabled in the next release, this will become less of a problem as all file transfers will be handled by a background process, and the main console window will be able to continue browsing the host system, adding files to the queue.

The Setup menu allows you to change the font used in displaying this and all other windows in the application, and also to set certain application options.

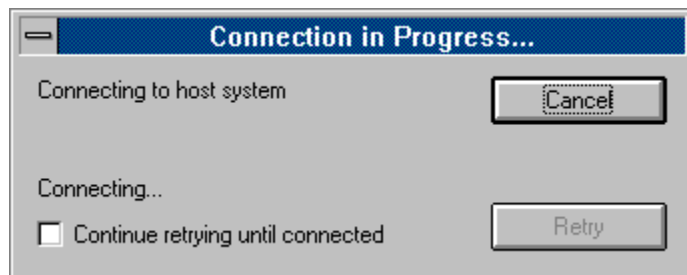
The View Logfile menu brings up a log of all FTP commands sent during the last session (within this run of the application). Whenever you connect to a system, this log is cleared, so it only retains information about the most recent session. This can be used in reporting bugs to the author. Copy the contents of the log into an Email and send it to mclouden@dfm.com.

Here is the main window - click on an area of the window to receive help.



The connection progress dialog that is displayed looks like this. The current phase of connecting is displayed along with the actual FTP commands that are being sent to the host system.

Once the connection is complete and you are logged into the system, the window is removed and replaced by the console window.



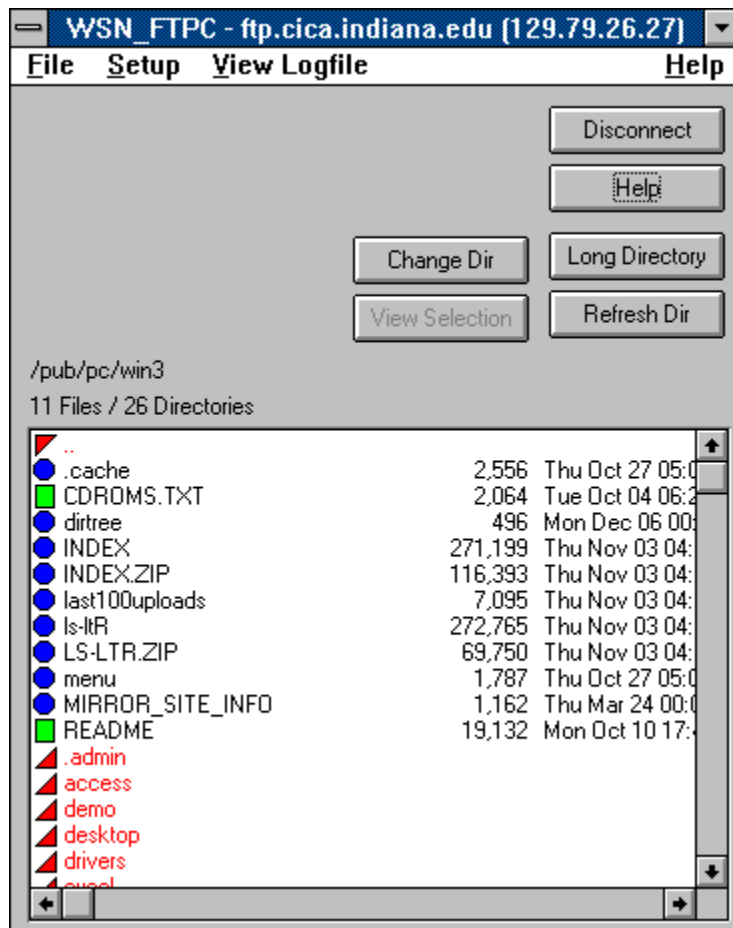
The console window displays the current directory and all files available in that directory.

To download a file, double-click on the appropriate entry in the listbox. To delete the file/sub-directory, or to create a new subdirectory, choose the File menu.

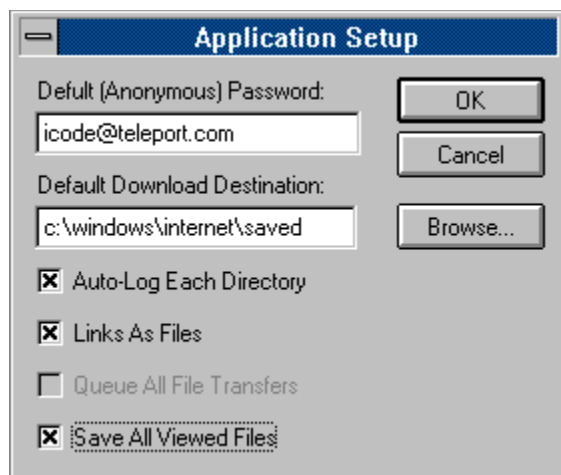
To send a file to the host system, again use the File menu. The File Open common dialog is displayed. This is modified with two checkboxes - the QueueTransfer checkbox (currently disabled) and the Send Binary checkbox. All transfers default to binary. To send a file as ASCII, clear the Send Binary checkbox.

To view a log of all commands sent to the host system during this session, use the View Logfile menu option.

To configure the application, use the Setup menu. Off of this menu, you can bring up a dialog to maintain your host configurations. Changes to the current configuration are saved, but do not effect the current session.



From the Setup menu, the application options dialog box can be displayed. For help on a particular configuration item, click on this display.



WSN_FNGR - Finger Client

SORRY!

Time is at a premium right now - and the Finger Client from the WSNWDEMO application has yet to be ported over to the new look found in this release.

The time required is minimal, but it just isnt there right now. Very soon, Finger will be back.

For WSNWDEMO Finger users:

An obvious bug exists in the WSNWDEMO version. You will lose data towards the end of the query. Well, I know you cant see that it has been fixed since you don't have the update, but it now works correctly. Look for an improved Finger client with this fix and a few enhancements in the next WSNWUTIL.

WSN_FNGS - A Finger Server

SORRY!

Time is at a premium right now - and the Finger server needs some cleaning up before release. It all works, but I want all the WSNWUTIL applications to look and act the same way. Right now the Finger server does not look or act like anything else. It will be available in the next WSNWUTIL release. Currently the Finger server provides information about the owner of the machine - i.e. a single plan file.

With the improvements seen in the new Windows NT 3.5, I have switched to this as my primary development environment. Consequently, the Finger server will support multiple user plan files. Multiple plans may or may not be found in the next WSNWUTIL release. If not this coming one, then the following one for sure.

Configuration

Each application in the collection has its own set of configuration items. All applications do though have the Font configuration item in common.

The Font menu item brings up the Font common dialog box. The font you select will be used in sizing the window and all the controls.

The resizing of a dialog box and all the controls included in the dialog is not a trivial operation. And as this is fairly new code, do not be too alarmed if the window being resized causes the display to be corrupted in some way. Exiting the application and restarting it will clear up the problem. The one known situation where the font selection may cause a problem is when the sizing of the window causes the menu bar to wrap to a second line. I have yet to figure any way around this except to limit the minimum font size you may select.

Options for all programs are saved in a single common INI file, named WSNETWRK.INI. **This file will be created and should reside in your Windows directory.** Some options may not be saved correctly if the file is stored anywhere else.

The library files (WSNET100.DLL, WSNWCONN.DLL and WSNWFTPC.DLL) should be available somewhere on your path, or in the same directory as the WSNWUTIL executables.

The help file WSNWUTIL.HLP should be copied into the same directory as the applications.

Select Host To Test

Standard drop-down listbox that shows all configured host names. Use **New Host** and **Delete Host** to maintain this list.

Test Interval

Specify the number of seconds to allow a single test to complete. If no response is received from the selected host in the number of seconds specified, WSN_CONN will test the host again, counting the expired test as an error in the statistics area. If a reply is received after the interval has expired, the error count is decremented and the late reply is counted towards the late reply count in the statistics area.

Even if the selected host does respond within the interval time period, WSN_CONN will not attempt the next test until the time has expired.

The appropriate interval rate depends on the speed of your network connection. Three seconds is the default length of time and should provide an accurate picture of your connection to the selected host.

The interval may be from 3 to 60 seconds.

A 60 second time-out is useful in that it will keep a slip connection alive (for those that have there connections timed out after a period of inactivity), while using up very little in system resources.

Echo Port Number

When using the TCP or UDP test method, WSN_CONN will communicate with the selected host via a specific port address. Normally that address is the standard port number 7 for both UDP and TCP tests. If you are testing a host that uses some other port besides 7, change the port number to match the host. ICMP testing does not require a port number.

To use the default (port number 7, or whatever you may have set in your services file), leave the port number at Zero.

Testing Protocol

Select the means by which WSN_CONN will test for connectivity to the specified host system.

UDP and TCP will be accepted by most machines. ICMP, while supported by all systems, is not always available. This depends on the vendor for your network software. As an example, the Microsoft TCP/IP does not allow ICMP testing.

If a dialog box jumps up stating that the address family is not supported, then you know that you may not use ICMP testing on your machine.

TCP is a connection based protocol. This means that a formal relationship will be created between your machine and the machine you are testing. This is a very reliable protocol, and the statistics generated by your test should reflect that reliability.

UDP is a connection-**less** protocol. Data sent to the host system is not guaranteed to be received at that host system. Your testing statistics will probably include a few errors. This is probably the most consistently supported testing method though.

New Host

Pressing this button will pop-up a dialog box. In this dialog box only input field you should enter the name of the host system you would like to add. If a name is not known, the systems numeric internet address may be supplied instead.

The named/numbered host will show in the host list listbox.

Delete Host

The currently selected host system (selected via the host list listbox) will be removed from the list.

Start/Stop

When first started, and whenever a test is not in progress, this button will show as Start. Pressing this button will begin testing the currently selected host. The button will then change to read Stop.

Pressing Stop will of course, stop the active test. The button is returned to Start.

Reset Stats

Press this to set all accumulated statistics back to zero. The statistics are automatically reset when a new test is begun.

Statistics

Whats a test without some numbers?

Attempts: How many times have we tested the current host **and the test completed OK**.

Late Replies: Number of attempts that were responded to after the interval time had expired.

Error Count: Number of attempts for which no response was received.

Average Time: In seconds, the average length of time between a sent test and the corresponding reply.

Minimum Time: In seconds, the shortest amount of time between a sent test and the corresponding reply.

Maximum Time: In seconds, the longest amount of time between a sent test and the corresponding reply.

For all time statistics, a late reply is not included in the calculations. These numbers are computed using successful tests only.

Current Status

Shows as Inactive when no test is in progress. During a test, the testing method and the host name are shown.

Help

Gets you to here.

Close/Exit

Exits the application after closing any open sockets/connections.

TCP/UDP Start and Stop

The ECHO server may accept TCP or UDP requests. Both types of service may be stopped or started individually.

When the application begins both TCP and UDP services are stopped. Pressing the appropriate Start button will begin the service and the button will be changed to read Stop. Pressing the button again will stop the service.

Note that stopping the TCP service will **not** close any active sessions.

TCP/UDP Port ID

WSN_ECHO will listen for incoming requests at a specific port number. That port number can be different for TCP and UDP. Specify the port numbers here.

To leave the service listening at the standard port for ECHO (port number 7), enter a 0 in these fields.

If you want the ECHO service to be available at more than one port number, WSN_ECHO can be run multiple times, each specifying there own port numbers. (If you run WSN_ECHO multiple times with the same port number, the most recently started server will receive all incoming request).

Max. Connects

Set a limit on the number of active TCP sessions allowable at any one time. Additional connection attempts will be rejected.

Statistics

WSN_ECHO maintains the number of TCP connections since being stated, the number of active TCP sessions, the number of TCP echos returned, and the number of UDP echos returned.

Reset Counts

Set all accumulated statistics back to zero.

Active TCP Connections

The number of currently connected TCP sessions.

For each session, a listbox item displays the address of the machine which is connected (sending the echos).

To disconnect a session, double click on its listbox entry. Or, select any number of sessions and select the Kill Selected Sessions button.

Kill Selected Sessions

Disconnects all selected sessions in the Active Sessions listbox.

Select Configuration

Choose a previously added configuration. On changing the selection, the stored parameters for this configuration are reflected in the rest of the window.

Host Name

The system host name or host address used by this configuration.

Host Port Number

The port number at which the host system accepts FTP connections. Usually this is 21. Leave the box at 0 to use this default.

User Name

The user name with which to log on to the host system.

Choose the **Use Anonymous** button to use the Anonymous user ID.

Password

Specify the password for the user name you will be logging on with.

If you are using the Anonymous login, then your password should be your full Email address. To use the default password (specified in the **Options** dialog box), click the **Use Default** button.

Account Information

If you must specify account information to get logged on, enter it here.

Remote Directory Name

Specify the name of the directory in the host system to which you will change every time you log on.

Local Directory Name

The default location for all downloaded or viewed files. Use the **Browse** button or edit this directly to change the location of these files. This may be different for every configuration.

Browse

Brings up a dialog box containing a directory listing. Use this to change to the directory to be used as the Local Directory for the current configuration.

Connect

Connect to the host named by the current configuration.

If all parameters have been filled in correctly and a configuration is selected then the window will be hidden and the Connection Progress window will be displayed.

Delete Config

Remove the current configuration.

New Config

Add a new configuration. Two items are needed here, a host name and a name to be assigned to the configuration.

Connect Status

Displays the progress of connection attempt.

All commands that get sent to the host system as part of the logging on process are displayed here as well.

Continue Retrying Until Connected

When a connection attempt fails, the user will be prompted to retry the connection or cancel back to the main window. When this checkbox is selected WSN_FTPC will automatically retry the connection. To stop retrying, just clear the checkbox.

A 1 second delay is imposed between connection failures and the following connection retry.

Cancel

Cancel the connection (either failed or in progress) and exit back to the main window.

Retry

Attempt to connect to the host system after a connection failure.

This is only enabled when a connection attempt has failed.

Directory Information

The current directory name is displayed along with the number of files and subdirectories found within it.

Directory Listing

Full listing of all files and subdirectories found in the current directory.

The key to this listing is as follows:

A **green square** represents an ASCII transferable file.

A **blue circle** represents a binary transferable file.

The **left triangle symbol** represents the parent directory.

The **right triangle symbol** represents a child directory.

Double-clicking on a file begins its transfer. If a DOS compatible filename cannot be derived from the way the server has the file named, the **File Save As** dialog box is displayed.

Double clicking on the parent directory entry will change to the parent directory. Double-clicking on a child directory will change to that directory.

To change the default transfer method from Binary to ASCII or ASCII to Binary, **Right Click** on the entry. The symbol will be changed. Note that if you change to ASCII format the [View Selection](#) button will be enabled, and if you change from ASCII to Binary, it will be disabled.

Keyboard equivalents to the double-click action are allowed. As is a shortcut for the Delete command. Pressing the Delete key is the same as selecting File|Delete. Hitting Return on a selected item is the same as double-clicking on it.

View Selection

If the currently selected file is an ASCII transferable file, the file will be downloaded and NOTEPAD will be run to display the file.

Change Directory

Allows you to enter a directory name and then switch to that directory.

Refresh Directory

Obtain a fresh file listing from the host system.

Long Directory Listing

View the directory listing as it was sent by the host system.

Disconnect

Close the current connection and return to the main window.

Disconnect will send a QUIT command to the server. There may be a small delay involved in its processing.

OK/Cancel

OK will save the changes you may have made and Cancel will discard them. Either returns you to the previous window.

Set Default (Anonymous) Password

Enter your Email address here. It can then be inserted in the host-configuration window whenever you select the **Use Anonymous** button.

Set Default Download Directory

Set the default local directory name. This will be used by all new configurations. Existing configurations can use the Use Default button above the Local Directory Name field.

Auto-Log each directory

If this is checked, then every time you change the remote directory, the new directory is logged and the file listing is updated.

If it is not selected, the directory listing is cleared and can be updated by using the Refresh Directory button.

Links As Files

If this is checked, any UNIX Symbolic Link files will be interpreted as pointing to a file. If it is not then Symbolic Link files will be interpreted as directories.

Save All Viewed Files

When a file is viewed, it can be downloaded to the local directory specified by the configuration or to the Windows TEMP directory.

TEMP files are not automatically cleaned up by the application (the file is given to NOTEPAD then promptly forgotten about by WSN_FTPC). If you do not automatically save these files, you should delete all VUE*.TMP files found in your Windows TEMP directory on a periodic basis.

If viewed files are automatically saved, they will be downloaded directly into your local directory.

Location of latest WSNWUTIL.ZIP release

Periodically, I upload WSNWUTIL to <ftp.cica.indiana.edu/pub/pc/win3/winsock>.

I **always** upload the absolute latest version of WSNWUTIL to RMII.COM/pub2/mclouden. I recommend getting WSNWUTIL file from RMII.

An information package for the WSNETWRK development library is available for download as INFODOC.ZIP in that same RMII directory.

