



Computers for Industry

## Documentation

### Ftp Client with Ethernet-Module EM01



## Table of contents

---

<b>Ftp Basics</b>	<b>4</b>
What is FTP?	4
What is a FTP Site?	4
What is a FTP Client?	4
<b>Ftp Servers</b>	<b>4</b>
Setting up your FTP Server	4
Serverlist	5
<b>Setting up a FTP Server on WinXP Professional</b>	<b>5</b>
Accessing an FTP site	5
Controlling Anonymous Access	5
Setting up an FTP site	6
Configuring the FTP site Controls	7
WinXP FTP Security Controls	8
<b>Third-party FTP software</b>	<b>10</b>
Setting up an FTP site	10
Configuring Serv-U	10
Creating FTP User Accounts	11
<b>Ftp and firewalls</b>	<b>12</b>
Software firewalls and FTP	12
Configure Zonealarm to allow FTP access	13
Configuring hardware firewalls for FTP	13
Ftp Security	13
<b>Introduction for data transfer</b>	<b>14</b>
Active mode	14
Passive mode	16
Binary vs. ASCII	18
<b>Getting started with the Tiger Basic Ftp Client</b>	<b>19</b>
<b>Settings</b>	<b>19</b>
Server	19
Client	19
Enter	19
Options	20
Necessary files	20
<b>Start</b>	<b>21</b>
Simple login / logout	21
Server code / expected code	22
<b>Data transfer</b>	<b>23</b>
Simple download	23
Simple upload	25
Get file list	27
Append data	28
Transfer in active mode	29
Restart	31
Representation type	31
Transfer mode	32

File structure	32
<b>File and directory operations</b>	<b>32</b>
Delete file	32
File size	32
Make new directory	33
Delete directory	33
Rename file or directory	33
Change working directory	33
Change to parent directory	33
Get working directory	34
<b>User rights</b>	<b>34</b>
<b>Other routines</b>	<b>35</b>
Help	35
No operation	35

# Ftp Basics

---

## What is FTP?

FTP (File Transfer Protocol) is the simplest and safest way to exchange files over the Internet. A site, called an FTP site, is used to transfer files back and forth. In order to transfer files from your computer (i.e., web pages that you have created) to the web, you will need to use FTP.

An FTP address looks a lot like a Website address except it uses the prefix *ftp://* instead of *http://*.

Website: <http://www.shsu.edu>

FTP site: <ftp://www.shsu.edu>

## What is a FTP Site?

An FTP site is a way to organize your files on your web space. You can create folders and folders within folders to keep everything in order.

FTP sites are generally password-protected, that is one would need a login name and password to connect. When uploading to web space, your login name would usually be your username on that site, or one you create for your own domain. For example, when you upload to your SHSU web space, your login name and password are the same as your Windows, E-Mail and Blackboard username and password.

To make an FTP connection you can use a standard Web browser or an FTP Client.

You can open your FTP connection through Internet Explorer and transfer your files by dragging and dropping them from your computer, but it is not recommended since it is difficult – sometimes impossible, and the transfers are unprotected.

When you connect with a client, file transfer is easier and safer. With a client, you can resume downloads that have not completed because of a disconnection, which is useful for dial-up users or times when your internet disconnects.

## What is a FTP Client?

An FTP Client is software which connects your computer to your FTP site and allows transfer of files back and forth between them.

An FTP Client usually has 2 panes. The left side represents the files on your computer, sometimes called the “Local” side. The right side contains any files and folders that are already on your web space, sometimes called the “Remote” side.

In example from WS\_FTP, transferring the files is as easy as selecting the file, and clicking the directional arrow to transfer from Local to Remote or vice versa. In some clients, you can also drag-and-drop the files from one side to the other.

Some FTP Clients have additional features such as file resuming, multiple file transfer, and queuing (setting up a queue order for files to be transferred).

# FTP Servers

---

## Setting up your FTP Server

Setting up an FTP server is quite simple. Some software has the ability to set passwords, directory access permissions, and a number of other FTP server functions. You can experiment with these attributes as you get more comfortable with your FTP server. For now, let's start with the basics.

To begin, you will need to set those directories you'd like to share, or 'host', on your FTP site. You can select an entire drive or any subdirectory of the computer you're using for your FTP server.

Second, you'll need to determine how people gain access to your FTP site. There are several options for granting permission to view, download and upload to your site.

Unlike most Websites, when a user visits an FTP site they must login. By default, a public FTP site uses an 'anonymous login'. Technically, this means the User Name is 'anonymous' and the Password is a user's e-mail address. However, often with an anonymous FTP site, the user is not required to enter any login information as these values are used by default.

It's important to keep in mind that anonymous logins should only be granted for viewing and downloading – you never want to allow public uploads to your FTP site. For uploading, you want to force a visitor to type in a unique User Name/Password in order to gain upload access to your FTP site. It's not uncommon for an FTP site to use a combination of anonymous and password protected logins.

Most FTP server software will detect the 'home IP' address of your computer during the FTP server setup or in the text display when the server becomes 'on-line'. As mentioned earlier, you will need this IP address when setting up a domain name for your FTP site.

### Serverlist

This list is not in any means complete. I have no intention to list all available subjects in this category. The intention is to show the most important and/or popular choices. In my opinion, FTP software should be free. The protocol is developed on a non-commercial basis, and free source code is available for any developer who wants to make a FTP server or client.

**Serv-U** Corporate edition during 30-day trial. Serv-U will continue to function indefinitely as the Personal edition.

<http://www.serv-u.com/customer/record.asp?prod=su>

**WU-FTPd** Freeware (GNU/GPL) The leading FTP server for UNIX

<http://www.wu-ftp.org/>

**G6 FTP** FTP server for Windows

<http://www.gene6.com/g6ftpd/>

**WFTPD** FTP server for Windows

<http://www.wftpd.com/>

**Proftpd** Popular FTP server for Linux/Unix systems

<http://www.proftpd.org/>

**Troll FTP Daemon** small, secure, multi-hosting FTP server for linux.

<http://www.troll.no/freebies/ftpd.html>

**War FTP Daemon** Freeware FTP server for Windows

<http://www.jgaa.com/warftpd.htm>

## Setting up a FTP Server on Windows XP Professional

### Accessing an FTP site using Windows XP and Internet Explorer.

Windows XP contains a built in FTP client, used through Internet Explorer, which you can use to access FTP sites as if they were directories on your computer. To do this, you simply need to enter the address of the FTP server into the address bar in Internet Explorer.

Let's take a closer look at a typical FTP address to see what it's made of:

FTP://67.68.255.65. This *example* address simply uses the IP address of the server computer, with the 'ftp://' at the start to inform Internet Explorer that it is looking to connect to an FTP site.

### Controlling Anonymous Access

FTP can also use DNS (Domain Naming System) addresses, as seen on the World Wide Web. For example: ftp://ftp.pcstats.com would make Internet Explorer attempt to connect to port 21 of the computer 'ftp' in the domain pcstats.com.

If you are connecting to an FTP site that has anonymous access disabled, meaning that you will have to enter a username and password to connect successfully, you must put your username into the address. For example: ftp://mike@67.68.255.65 or ftp://mike@ftp.pcstats.com

Assuming the username is correct, a password window will open so you can authenticate yourself and then enter the FTP site.

If the FTP site you are trying to connect to uses an alternate port instead of the default port 21, you will also have to specify this. For example, if the server were using port 1056 you would enter: FTP://67.68.255.65:1056 or ftp://mike@ftp.pcstats.com:1056

Essentially, FTP addresses can be entered into the IE address bar just as you would WWW addresses, with the only catch being that you must put the ftp:// before the rest of the address, otherwise Internet Explorer will assume that you are trying to connect to a website and not an FTP server. Websites use port 80 by default.

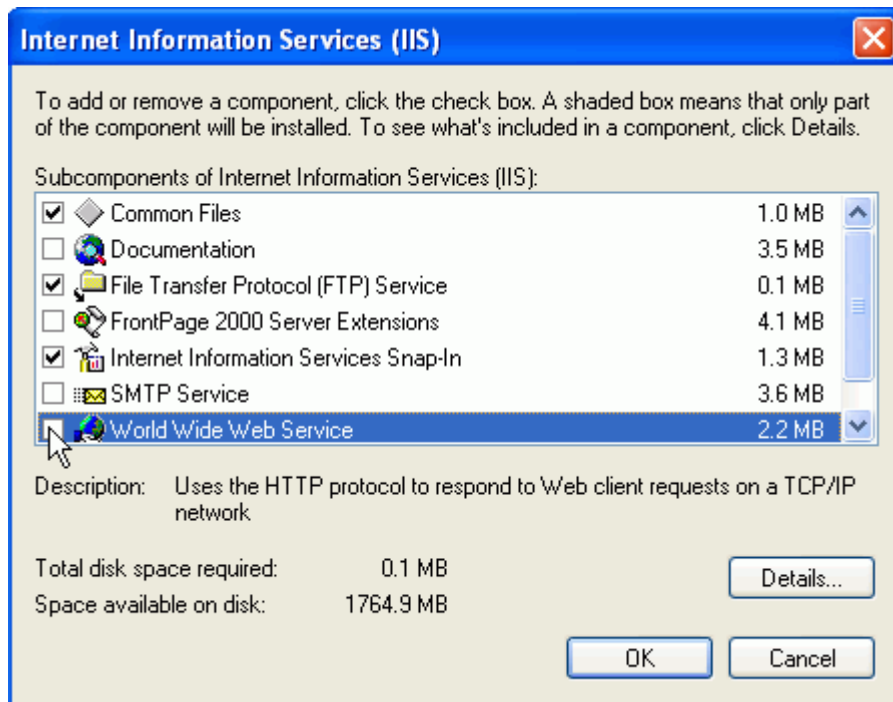
Once you have connected to the FTP site, you are presented with a directory window of its contents, which you can manipulate as if it was a directory on your local computer (subject to the permissions you have in the FTP site, of course). You can open files, copy and paste into your other directories, and copy from your computer to the FTP site if you have write permission.

### **Setting up an FTP site Using Windows XP Professional**

Windows XP professional (as well as Windows 2000) includes Microsoft's IIS (Internet Information Server) which can be used to create an FTP site on your computer. It's a fair bit less complicated and less flexible than using some third-party FTP server software packages, so we will give you guides for setting up both. If you are using XP Home you will need to use third-party software. There is no way to publish an FTP site with the Home Edition of XP.

The first step is to check that IIS (Internet Information Services, Microsoft's web-server application) is configured properly.

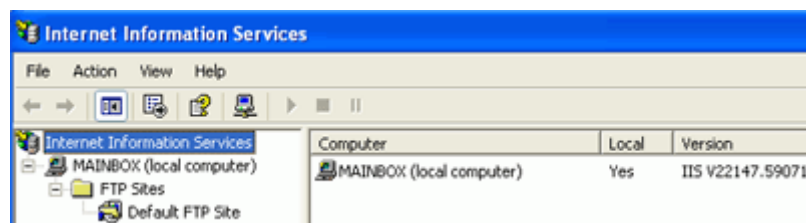
Go to start\control panel\add/remove programs\ choose the 'add/remove windows components' button from the bar on the left. Highlight the item 'Internet information services (IIS)' If it is unchecked, check it, then click 'details.'



The components you will need are: 'common files,' 'file transfer protocol (FTP) service' and 'internet information services snap-in.' Uncheck any others then click next. IIS will configure itself, and you may be prompted for the XP CD.

### Configuring the FTP site Controls

After IIS has been installed, an FTP site is automatically created for the directory 'c:\inetpub\ftproot.' Of course, this directory is currently empty. It is also *completely unsecured*, allowing anyone who enters ftp://(your IP address) in their browser or FTP client to connect to your computer. Next step is to configure your new site.



'internet information services.'

Go to start\control panel and select the 'switch to classic view' option in the upper left corner. From the classic control panel window, select 'administrative tools,' then

From here, expand '(local computer)' and 'FTP sites' until you have 'default FTP site' in the left hand pane. Right click on 'default FTP site' and select rename if you would like it to be called something a bit catchier. After all, it's your site now.

Now, right click on your site and select 'properties.'

This window is the life-blood of your FTP site. Let's get familiar with it. The first tab, 'FTP site,' allows you to rename the site, set the port through which users can connect (leave it at 21 for now), set connection and logging information and view who is currently connected to your FTP site.

The screenshot shows the 'Default FTP Site Properties' dialog box with the 'FTP Site' tab selected. The 'Identification' section contains a 'Description' field with 'Default FTP Site', an 'IP Address' dropdown set to '(All Unassigned)', and a 'TCP Port' field set to '21'. The 'Connection' section has radio buttons for 'Unlimited' and 'Limited To:' (selected), with a value of '10' connections and a 'Connection Timeout' of '900' seconds. The 'Enable Logging' checkbox is checked, and the 'Active log format' is set to 'W3C Extended Log File Format'. There are buttons for 'Properties...', 'Current Sessions...', 'OK', 'Cancel', 'Apply', and 'Help'.

The connection section of this tab has two parts, the 'limited to:' box sets the maximum amount of users that can connect to you FTP site at the same time. Note that with XP Professional, the maximum is always 10 concurrent users. You can set this to less if you'd like.

### WinXP FTP Security Controls

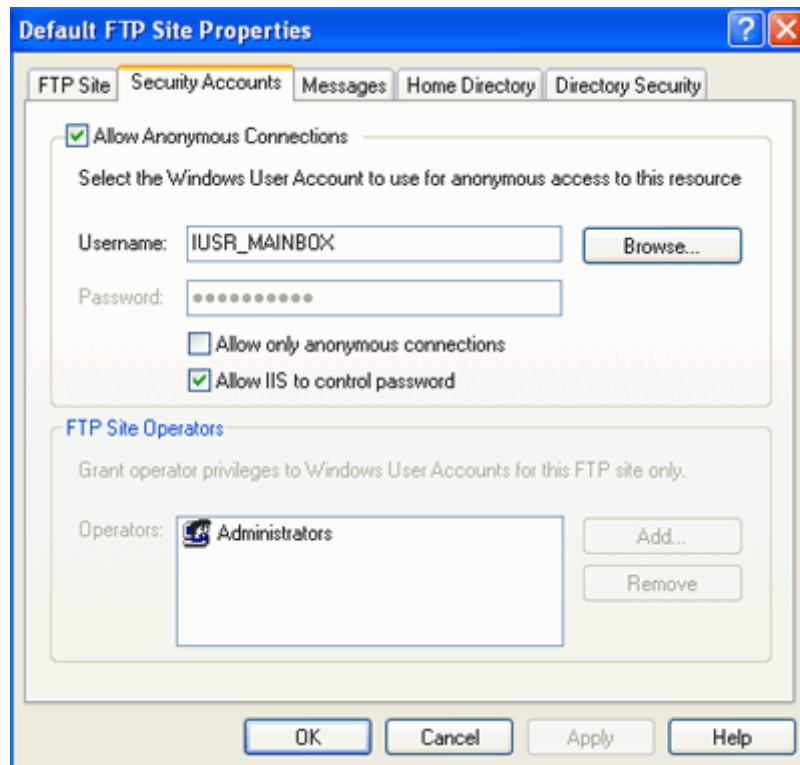
The screenshot shows the 'FTP User Sessions' dialog box. It contains a table with columns 'Connected Users', 'From', and 'Time'. The table lists one user: 'IEUser@' from '67.68.52.158' connected for '0:01:08'. Below the table, it says '1 User(s) Currently Connected.' and has buttons for 'Disconnect' and 'Disconnect All'. On the right side, there are buttons for 'Close', 'Refresh', and 'Help'.

Connected Users	From	Time
IEUser@	67.68.52.158	0:01:08

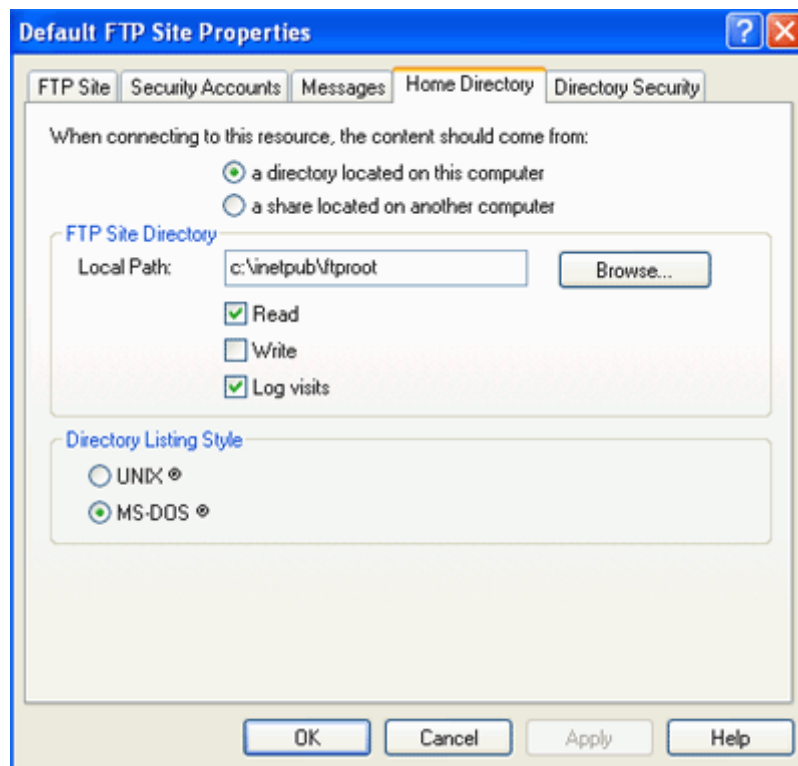
The 'connection timeout' box shows the amount of time a connected user will be allowed to remain idle before being disconnected. By clicking the 'current sessions' button at the bottom, you can view who is currently connected to your FTP site, and if you wish, disconnect them.

The next tab 'security accounts,' controls whether anonymous users (that means everyone) are allowed to access your FTP site or not. As mentioned above, by default anyone can access your FTP site without a username or password. IIS uses a built-in user account with a defined set of restrictions to authenticate anyone who connects. This user account, the 'IUSR\_(computer name) account, is created when IIS is installed, and is also used to allow access to websites you may publish. It is restricted from accessing non-IIS parts of your Windows system.





To be honest, there is not really a correct choice for this setting. If you allow anonymous access, anyone can connect to your FTP site and view any files that you place there. Disabling anonymous access has its own set of risks, however, which we will cover in the 'FTP security' section below. For now, leave anonymous access enabled. The next section, 'messages,' simply allows you to set various text messages which users connecting to your site will see. Fairly self-explanatory.



The fourth tab, 'home directory,' allows you to configure which directory (folder) in your system will be accessed by the FTP site. In the 'FTP site directory' section, you can choose this directory, and designate whether connected users will have permission to write to and/or read from the site, and whether their visits will be logged.

Choose the directory you wish to share files from, or leave it at the default and simply copy files you wish to make available into the directory using explorer.

## Third-party FTP software

### Setting up an FTP site with third-party software

Since many PCstats readers may be using XP Home or Windows 9x/ME which do not include IIS and thus cannot be used to create FTP sites, we thought we'd run through creating an FTP server using third-party software. In this case we've chosen the popular Serv-U program by RhinoSoft.

We chose Serv-U because its personal edition is free for non-commercial use, and it is quite easy to grasp for neophyte users. Serv-U offers some additional security and flexibility over the IIS implementation of FTP, at least with Windows XP. Let's look at setting it up...

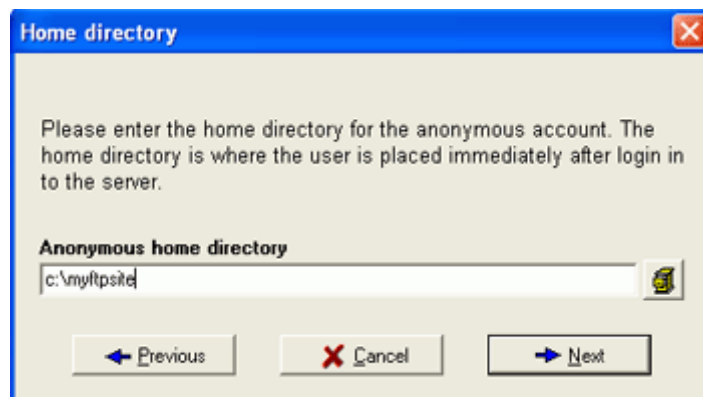
Once you have downloaded and installed the software, start it up. The setup wizard will run. Press 'next' three times to start the FTP server. You will be prompted for your IP address. Leave it blank. Press 'next.'

You are asked to name your 'domain' (Serv-U's name for your FTP site). Choose whether you wish Serv-U to start automatically when you boot Windows, or to start only when you run the program from the desktop.

### Configuring Serv-U

The next screen brings the first major difference between Serv-U and Microsoft's IIS. You are asked whether you wish to allow anonymous access, meaning that anyone will be able to log into your FTP site by using a special 'anonymous' user account created for this purpose.

The difference here is that the anonymous account created resides only within the Serv-U



program, as do all other accounts you will create for accessing this FTP site. Separating Windows user accounts from the accounts you create to access the FTP site adds a layer of security. If you do not choose to use anonymous access, you will have to create user accounts within Serv-U with permission to access your site. More on this in a moment...

If you elected to allow anonymous access, you will be prompted for a directory, which will serve as the 'home base' for anonymous users. When they connect, they will see the directory you specify here first.

Once you enter the directory you will be asked if you wish to limit anonymous users to this directory only, or allow them to browse through to other directories. This brings up the second major difference between Serv-U along with most other third-party FTP servers and the Windows implementation of FTP: you are not limited to a single directory.

Of course, you may want to be limited to a single directory, as it makes keeping a handle on things much simpler, but we digress. For the time being, choose to lock anonymous users into the directory you specified.



You will now be prompted to create 'named accounts' which are user accounts with passwords analogous to those seen in Windows, except that these are used only for FTP access within Serv-U.

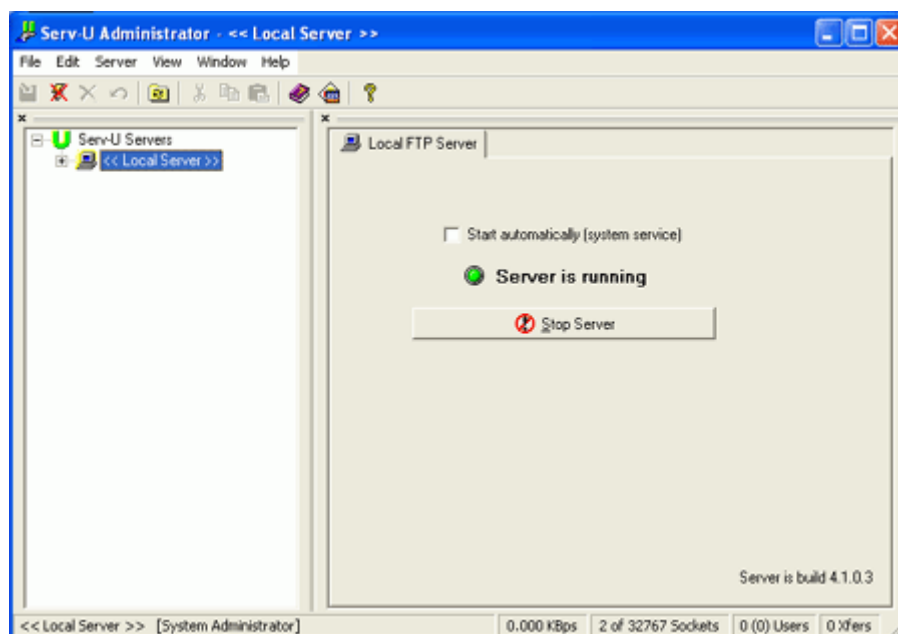
For the time being, create a named account and password of your choice and give that user a different initial directory than the one you previously assigned to anonymous users. When prompted,

choose not to lock the named user into his home directory. The final question the setup wizard will ask is whether you wish to give the user you just created any administrative privileges, allowing him to configure the FTP site remotely. We will answer 'no' to this one for a simple reason. Remote Access is disabled in the 'personal' edition of Serv-U.

The version you are using is the evaluation version which contains all the features of the Professional edition, but reverts to the personal edition if not purchased within 30 days. This tutorial is based around the features available in the personal edition.

### Creating FTP User Accounts

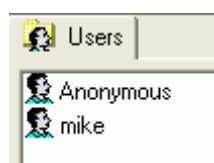
Once you have completed the setup wizard, you will be presented with the full Serv-U window. Your FTP site is now up and running. Test it from another computer using the method listed in the first section.



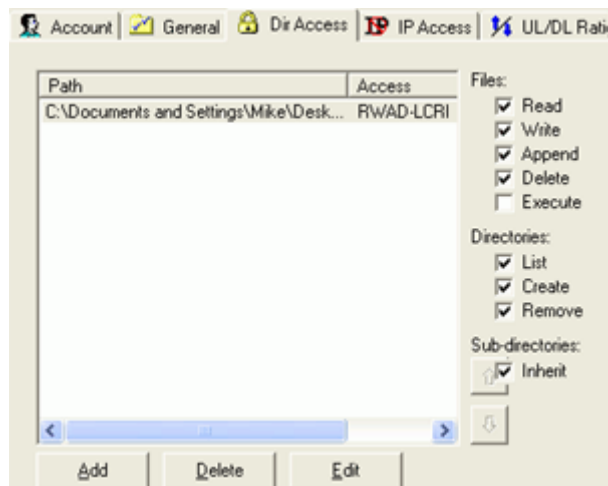
As you might notice, it's a good deal more complicated than the Windows implementation of FTP. The first thing we will do here is expand 'local server' and 'domains' until you can see the domain that you created. Expand that too.

While we don't have the space to go over all the options available to you in this program, we will cover a few

important ones. For more help, consult the Serv-U help files or their website. First, in the 'settings' menu, go to the 'IP access' tab. This section allows you to block or allow individual computers to access your site based on their IP address.



The 'activity' option allows you to view users connected to your domain. By right clicking on a connected user, you can send a message to him or her, disconnect them, stop their data transfers or even eavesdrop on the commands they are sending to your server. The 'users' option contains the user accounts you have created within Serv-U.



You will notice that the anonymous account is here, along with the named account that you created. Select the named account in the left-hand pane. From the user properties menu, you have several options: From the 'accounts' tab you can disable users and change their home directories.

The 'directory access' tab is extremely important, as it controls the rights this user will have once he is connected to your FTP site. For example, if you only wish clients to be able to read and copy files from your FTP site, give them the 'read' file

permission and the 'list' directory permission. If you want them to be able to add and edit files, you must assign the 'write' and 'append' file permissions, etc.

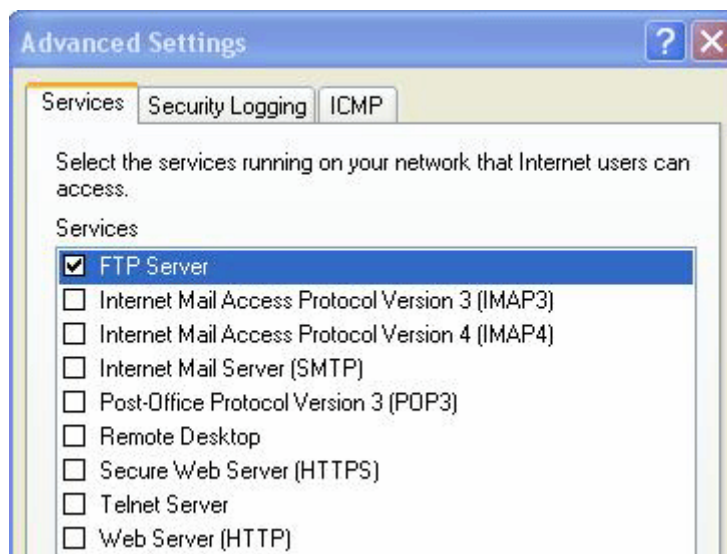
This gives you a good starting idea of how to use Serv-U to set up your own FTP site. Be aware that the version you are using will revert to the personal version after 30 days. The limitations of the personal version are: 1 domain only, maximum of 5 users, one concurrent connection only, and no encryption. None of these are a problem if you wish to create a site to enable you to access your files remotely or allow a friend to download from your system.

## FTP and firewalls

If you use some form of hardware or software firewall to protect your computer, you will probably need to do a little more work to get FTP to operate correctly.

### Software firewalls and FTP

The two most common software firewalls are the built-in Windows XP firewall and Zonealarm by Zone Labs. To configure the *Windows XP firewall* to allow FTP access: Go to start\control panel\network connections, right click on the icon for your Internet connection and select 'properties.'



Go to the 'advanced' tab and click the 'settings' button to configure your firewall (ensure that the firewall is enabled first; if it is enabled there will be a check in the 'protect my computer...' box).

From the 'services' tab, simply place a checkmark in the 'FTP server' box. This will allow FTP traffic on port 21 to enter your computer. Press 'ok.'

## How to configure Zonealarm to allow FTP access

Programs ▲	Access		Server	
	Trusted	Internet	Trusted	Internet
Files and Settings 1	?	?	?	?
FTP Serv-U Adminis	✓	✓	✓	✓
Generic Host Proce	✓	✓	✓	✓
HyperSnap-DX	?	?	?	?
InstallShield (R) Set	?	?	?	?
Internet Explorer	✓	✓	?	?
Internet Information	✓	✓	✓	✓

From the main Zonealarm window, select 'program control.'

If you are using Window's built in FTP server, you need find the entry for 'internet information services' and place checkmarks next to 'access\internet' and 'server\internet.'

If you are using Serv-U or some other third party program, locate the program on the list (if it is not present, click 'add' and browse to the program's executable file to add it to the list) and again place checkmarks next to 'access\internet' and 'server\internet.'

This will allow your FTP site to send and receive information through the Zonealarm firewall.

## Configuring hardware firewalls for FTP

Home Internet sharing devices like Cable/DSL routers are very common, and almost all come with some form of firewall that is enabled by default. To successfully pass FTP traffic through these devices, you will need to create a 'virtual server' entry in the setup of your Internet sharing device. Pictured below is an example of this from an SMC Barricade home DSL/cable router.

Enter the information you want.

	Private IP	Private Port	Type	Public Port
1.	192.168.5.219	21	<input checked="" type="radio"/> TCP <input type="radio"/> UDP	21
2.	192.168.5.		<input checked="" type="radio"/> TCP <input type="radio"/> UDP	
3.	192.168.5.		<input checked="" type="radio"/> TCP <input type="radio"/> UDP	

A virtual server is an instruction to your Internet sharing device telling it to forward any traffic it receives on a specified port to a specific computer inside your network. For example,

if you create a virtual server for port 21, IP address 192.168.5.220, your internet sharing device will listen for traffic coming in on port 21, then pass that traffic through the firewall to the computer with that IP address.

Though the instructions will vary depending on the brand of your device, what you will need to do is find the 'virtual server' setup section (or equivalent), and specify the IP address of the computer that is running the FTP server (to find this, go to start\run and type 'cmd' then 'ipconfig.'). You will need to enter port 21 for data coming into and out of the router.

Once this is saved, FTP information will be able to pass through your firewall.

## FTP security

Important topic. The problem with FTP is that, by default, it is an extremely insecure protocol. Usernames and passwords are not encrypted in any way when they are sent from the client to the server, and so are prime targets for anyone intercepting network packets between your server and your clients.

This is the reason that the Windows FTP server software recommends that you use only anonymous access for your FTP site, as the alternative is to use valid user accounts from your XP installation.

If these credentials are intercepted, they could be used to severely compromise the security of your entire system, never mind your FTP site. Hence the recommended practice for home users is to allow anonymous access to the FTP site directory and simply not place sensitive files there. Obviously, this is not going to meet everyone's needs, so there are alternative methods of securing FTP transactions.

Generally speaking, these involve using SSL (Secure Socket Layer) or some other encryption method to encrypt the plain FTP information, creating a secure channel between the client and server. For more information on SSL and other methods of encryption, see PCstats' Beginners Guide to encryption [here](#).

Most third-party FTP server software packages support encryption as part of the FTP program itself, but using IIS for Windows XP, the only possible method of security is to use a method that encrypts all traffic between the server and a specific client, such as a VPN (Virtual Private Network).

Serv-U supports creating an SSL certificate within the program for encrypting traffic, but only in their commercial versions of the program. The free personal edition does not have this feature.

So to sum up, unless you have specifically placed security measures, assume that all FTP traffic is inherently insecure. Therefore, don't put data in your FTP site that you would not want seen by the general public. Don't be scared away from it though, since the fact that anyone can access your FTP site does not affect the security of the rest of your system unless you are using your Windows user accounts with IIS.

## Introduction for data transfer

---

One of the most commonly seen questions when dealing with firewalls and other Internet connectivity issues is the difference between active and passive FTP and how best to support either or both of them. Hopefully the following text will help to clear up some of the confusion over how to support FTP in a fire walled environment.

FTP is a TCP based service exclusively. There is no UDP component to FTP. FTP is an unusual service in that it utilizes two ports, a 'data' port and a 'command' port (also known as the control port). Traditionally these are port 21 for the command port and port 20 for the data port. The confusion begins however, when we find that depending on the mode, the data port is not always on port 20.

So there are two ways to transfer data via ftp:

- in *ACTIVE* mode
- in *PASSIVE* mode

### Active FTP

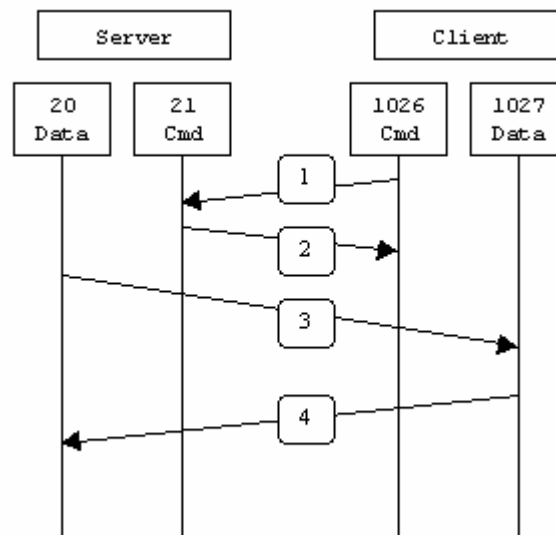
In active mode FTP the client connects from a random unprivileged port ( $N > 1024$ ) to the FTP server's command port, port 21. Then, the client starts listening to port  $N+1$  and sends the FTP command **PORT N+1** to the FTP server. The server will then connect back to the client's specified data port from its local data port, which is port 20. From the server-side firewall's standpoint, to support active mode FTP the following communication channels need to be opened:

- FTP server's port 21 from anywhere (Client initiates connection)
- FTP server's port 21 to ports  $> 1024$  (Server responds to client's control port)

FTP server's port 20 to ports > 1024 (Server initiates data connection to client's data port)

FTP server's port 20 from ports > 1024 (Client sends ACKs to server's data port)

When drawn out, the connection appears as follows:



In step 1, the client's command port contacts the server's command port and sends the command PORT 1027. The server then sends an ACK back to the client's command port in step 2. In step 3 the server initiates a connection on its local data port to the data port the client specified earlier. Finally, the client sends an ACK back as shown in step 4.

The main problem with active mode FTP actually falls on the client side. The FTP client doesn't make the actual connection to the data port of the server--it simply tells the server what port it is listening on and the server connects back to the specified port on the client. From the client side firewall this appears to be an outside system initiating a connection to an internal client--something that is usually blocked.

### Example

Below is an actual example of an active FTP session. The only things that have been changed are the *server names*, *IP addresses*, and *user names*. In this example an FTP session is initiated from testbox1.slacksite.com (192.168.150.80), a linux box running the standard FTP command line client, to testbox2.slacksite.com (192.168.150.90), a Linux box running ProFTPd 1.2.2RC2. The debugging (-d) flag is used with the FTP client to show what is going on behind the scenes.

Client prompts and information are black. Everything in **red** is the debugging output which shows the actual FTP commands being sent to the server and the responses are **blue** generated from those commands. Transferred data is shown in **green**, and user input is in **bold**.

There are a few interesting things to consider about this dialog. Notice that when the PORT command is issued, it specifies a port on the *client* (192.168.150.80) system, rather than the server. We will see the opposite behavior when we use passive FTP. While we are on the subject, a quick note about the format of the PORT command. As you can see in the example below it is formatted as a series of six numbers separated by commas. The first four octets are the IP address while the second two octets comprise the port that will be used for the data connection. To find the actual port multiply the fifth octet by 256 and then add the sixth octet to the total. Thus in the example below



the port number is  $(14 \times 256) + 178$ , or 3762. A quick check with 'netstat' should confirm this information.

```
testbox1: {/home/p-t/slacker/public_html} % ftp -d testbox2
Connected to testbox2.slacksite.com.
220 testbox2.slacksite.com FTP server ready.
Name (testbox2:slacker): slacker
---> USER slacker
331 Password required for slacker.
Password: TmpPass
---> PASS XXXX
230 User slacker logged in.
---> SYST
215 UNIX Type: L8
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
ftp: setsockopt (ignored): Permission denied
---> PORT 192,168,150,80,14,178
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for file list.
drwx----- 3 slacker  users      104 Jul 27 01:45 public_html
226 Transfer complete.
ftp> quit
---> QUIT
221 Goodbye.
```

## Passive FTP

In order to resolve the issue of the server initiating the connection to the client a different method for FTP connections was developed. This was known as passive mode, or PASV, after the command used by the client to tell the server it is in passive mode.

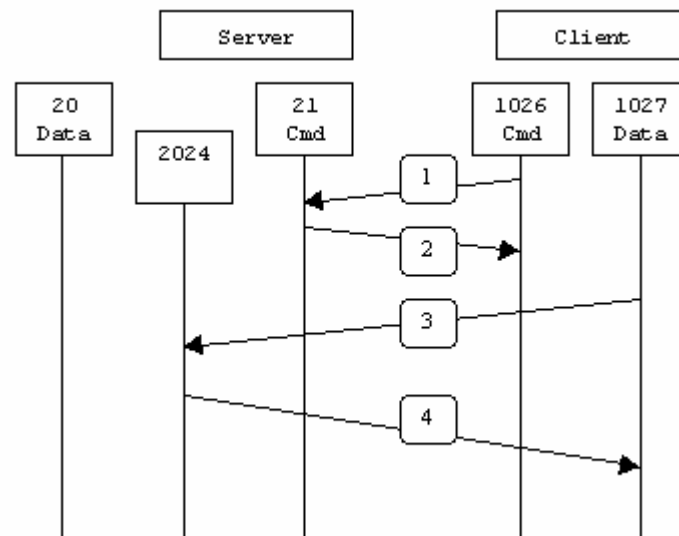
In passive mode FTP the client initiates both connections to the server, solving the problem of firewalls filtering the incoming data port connection to the client from the server. When opening an FTP connection, the client opens two random unprivileged ports locally ( $N > 1024$  and  $N+1$ ). The first port contacts the server on port 21, but instead of then issuing a PORT command and allowing the server to connect back to its data port, the client will issue the PASV command. The result of this is that the server then opens a random unprivileged port ( $P > 1024$ ) and sends the PORT P command back to the client. The client then initiates the connection from port  $N+1$  to port P on the server to transfer data.

From the server-side firewall's standpoint, to support passive mode FTP the following communication channels need to be opened:

- FTP server's port 21 from anywhere (Client initiates connection)
- FTP server's port 21 to ports  $> 1024$  (Server responds to client's control port)
- FTP server's ports  $> 1024$  from anywhere (Client initiates data connection to random port specified by server)
- FTP server's ports  $> 1024$  to remote ports  $> 1024$  (Server sends ACKs (and data) to client's data port)



When drawn, a passive mode FTP connection looks like this:



In step 1, the client contacts the server on the command port and issues the PASV command. The server then replies in step 2 with PORT 2024, telling the client which port it is listening to for the data connection. In step 3 the client then initiates the data connection from its data port to the specified server data port. Finally, the server sends back an ACK in step 4 to the client's data port.

While passive mode FTP solves many of the problems from the client side, it opens up a whole range of problems on the server side. The biggest issue is the need to allow any remote connection to high numbered ports on the server. Fortunately, many FTP daemons, including the popular WU-FTPD allow the administrator to specify a range of ports which the FTP server will use.

The second issue involves supporting and troubleshooting clients which do (or do not) support passive mode. As an example, the command line FTP utility provided with Solaris does not support passive mode, necessitating a third-party FTP client, such as ncftp.

With the massive popularity of the World Wide Web, many people prefer to use their web browser as an FTP client. Most browsers only support passive mode when accessing ftp:// URLs. This can either be good or bad depending on what the servers and firewalls are configured to support.

### Example

```
testbox1: {/home/p-t/slacker/public_html} % ftp -d testbox2
Connected to testbox2.slacksite.com.
220 testbox2.slacksite.com FTP server ready.
Name (testbox2:slacker): slacker
---> USER slacker
331 Password required for slacker.
Password: TmpPass
---> PASS XXXX
230 User slacker logged in.
---> SYST
215 UNIX Type: L8
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> passive
Passive mode on.
```

```
ftp> ls
ftp: setsockopt (ignored): Permission denied
---> PASV
227 Entering Passive Mode (192,168,150,90,195,149).
---> LIST
150 Opening ASCII mode data connection for file list
drwx----- 3 slacker  users      104 Jul 27 01:45 public_html
226 Transfer complete.
ftp> quit
---> QUIT
221 Goodbye.
```

## Summary

The following chart should help admins remember how each FTP mode works:

```
Active FTP :
    command : client >1024 -> server 21
    data     : client >1024 <- server 20

Passive FTP :
    command : client >1024 -> server 21
    data     : client >1024 -> server >1024
```

A quick summary of the pros and cons of active vs. passive FTP is also in order:

Active FTP is beneficial to the FTP server admin, but detrimental to the client side admin. The FTP server attempts to make connections to random high ports on the client, which would almost certainly be blocked by a firewall on the client side. Passive FTP is beneficial to the client, but detrimental to the FTP server admin. The client will make both connections to the server, but one of them will be to a random high port, which would almost certainly be blocked by a firewall on the server side.

Luckily, there is somewhat of a compromise. Since admins running FTP servers will need to make their servers accessible to the greatest number of clients, they will almost certainly need to support passive FTP. The exposure of high level ports on the server can be minimized by specifying a limited port range for the FTP server to use. Thus, everything except for this range of ports can be firewalled on the server side. While this doesn't eliminate all risk to the server, it decreases it tremendously.

## Binary vs. ASCII

Files found at Anonymous FTP sites can either be stored in ASCII mode (pure text files usually ending with either .txt or .asc) or non-text files stored in binary mode (also known as image mode [pictures, software, music, speech, word, etc.]). ASCII mode automatically adjusts the file during the transfer so that the file is a valid text file when it is stored on the receiving end. A binary file is left alone and transferred verbatim. Before downloading be sure to set the transfer mode to the proper type by typing or selecting **ascii** or **binary** prior to the **get** or **mget** command. The FTP site should respond with *type set to A* or *type set to I* acknowledging your request.

If you accidentally transfer a binary file with the transfer mode set to ASCII, the file received will be corrupted and useless. However, the reverse will work fine. If you don't know the type of a file, set the transfer mode to binary. Then you're sure that your files will always transfer correctly.

In the following example, we will get the binary zip file tterm23.zip (Terra Term Telnet Program) from an anonymous FTP session.

```

ftp> binary
200 Type set to I.
ftp> cd /pub/ibm
250-Please read the file README
250- it was last modified on Fri Aug 15 10:59:50 1997 - 1112 days ago
250 CWD command successful.
ftp> get tterm23.zip
200 PORT command successful.
150 Opening BINARY mode data connection for tterm23.zip (943376 bytes).
226 Transfer complete.
local: tterm23.zip remote: tterm23.zip
943376 bytes received in 0.072 seconds (12744.91 Kbytes/s)
ftp> quit

```

The files are now in your account. This file is in a compressed format, so it will have to be uncompressed before it can be used.

## Getting started with the Tiger ftp-client

---

### Settings

#### Server

If your ftp server was installed, is running successfully and has an user account, note the following parameters:

- Ip address
- Command port
- Username
- Password

#### Client

Choose a local ip address for the ethernet module and set the subnet mask.

If the module is connected to a local network, set the standard gateway to communicate with the internet.

Optional you can choose two local ports. One for the command connection and the other for the data connection.

#### Enter

Now you can enter these settings into "*ftp\_settings.inc*":

```

#define DEFAULT_GATEWAY      0C0A801FDh
#define LOCAL_IP_ADDRESS     0C0A8012Eh
#define LOCAL_IP_SUBNET_MASK 0FFFFFF00h

```

These settings must be given in hexadecimal code (two-digit for each octet), leading with "0" and ending with "H". To convert your numbers you can use the [dec\\_hex\\_converter.html](#).

```

#define LOCAL_COMM_PORT      10000
#define LOCAL_DATA_PORT      10001

```

The port numbers are useful, if your module is behind a firewall. Don't forget to change the settings of your firewall. The firewall must refer incoming data with these ports to your client ip.

```

#define MAX_NUM_DATA_CONNECTIONS 20
#define MAX_NUM_FTP_SESSIONS     5
#define RECIEVE_TIME_OFFSET      500

```

```
#define TIME_OUT_DATA_RECEIVE 30000
```

`MAX_NUMDATA_CONNECTIONS` and `MAX_NUM_FTP_SESSIONS` sets the number of sockets, needs to be opened.

`RECIEVE_TIME_OFFSET` is the time in ms, the client waits for one server response packet. `TIME_OUT_DATA_RECEIVE` is the time in ms, the server waits for the total server response.

```
#define FTP_IP_ADDRESS      0c0a8012ah
#define FTP_COMMAND_PORT   "21"
#define FTP_USERNAME       "ftpUser"
#define FTP_PASSWORD       "ftpPass"
```

I think these settings are self-explained.

After that, we can approach to the first ftp session with the ethernet module.

### Options

You can set the following options, by comment out.

```
#define SET_LOCAL_DATA_PORT
#define SET_LOCAL_COMM_PORT
```

If these lines are enabled, the command requests and responses will be transferred over port `LOCAL_COMM_PORT` and over port `LOCAL_DATA_PORT` set in `"ftp_settings.inc"`.

If you want to transfer data in active mode, this data port number will be ignored, because you have to set the port in the current subroutine. More about this later.

```
#define PRINT_STATUS_MSG
```

You can print the status of each executed subroutine. Either it shows the error location and the last error code for a better error analysis, if something was wrong or a success message. Define the device number with:

```
#define DISPLAY_DEV_NUM    #DEV_NUM
```

All Status messages will be printed on device number `#DEV_NUM`, now.

```
#define MAKE_FTP_LOG
```

The sent request and retrieved answers, will be saved in `"sFtpLog$"`. So you can better comprehend what happened between server and module.

```
#define RECV_DATA_IN_LOG
#define SENT_DATA_IN_LOG
```

Sent and retrieved data can be stored in `"sFtpLog$"`, too.

If you want to store the ftp log data in flash, use the following subroutine:

```
call putIntoFlash(sFtpLog$)
```

The size of written flash data will be saved in `lFlashBufferSize`.

### Necessary files

Include the following files:

```
define_a.inc
ufunc3.inc
ftp_needed_files.inc
```

---

## Start

### Simple login/logout

Now its time to start a ftp session. At first, the sockets must be initialized, the local ip and the gateway must be set with:

```
call prepareClient(blError)
```

If this subroutine was successfully executed, `blError` is `FALSE`.

Now you can login to your ftp server. Make sure, that the server is running.

```
call ftpLogIn(FTP_IP_ADDRESS, FTP_COMMAND_PORT, FTP_USERNAME, &
              FTP_PASSWORD, blError)
```

The client tries to connect to `FTP_IP_ADDRES` on port `FTP_COMMAND_PORT` for outgoing requests and incoming answers. If connection is build up, it tries to login with `FTP_USERNAME` and `FTP_PASSWORD`. If login was successful, `blError` is `FALSE` and you can see the user on the ftp server application.

Now you already log out:

```
call ftpQuitSession(blError)
```

`blError` is `FALSE` if you are logged out.

### Total code

```
`#define SET_LOCAL_DATA_PORT
`#define SET_LOCAL_COMM_PORT

#define PRINT_STATUS_MSG
#define DISPLAY_DEV_NUM

#define MAKE_FTP_LOG

`#define RECV_DATA_IN_LOG
`#define SENT_DATA_IN_LOG

#include define_a.inc
#include ufunc3.inc
#include ftp_needed_files.inc
```

#### Task Main

```
BYTE blError

`Install graphic display driver with device number DISPLAY_DEV_NUM
`install serial device driver

call prepareClient(blError)
if blError = FALSE then
    call ftpLogIn(FTP_IP_ADDRESS, FTP_COMMAND_PORT, FTP_USERNAME, &
                  FTP_PASSWORD, blError)

    if blError = FALSE then
        call ftpQuitSession(blError)
    endif
endif

call putIntoFlash(sFtpLog$)
```

End

### Display output

If you didn't change the "LANG\_ENG.INC", the display shows:

```
Status: ☺ successfully executed – ftpLogin()
Status: ☺ successfully executed – ftpQuitSession()
```

### Ftp Log

The flash has the following entries:

The server responses are **blue** and the client requests are **red**.

220 FTP Test Server ready

USER ftpUser

331 Password required for ftpUser.

PASS ftpPass

230 User ftpUser logged in.

QUIT

221 Bye bye ...

### Server code/ Expected code

In the ftp log you can see the server code. It is the leading three figure code. All further ftp subroutines need an expected code. That's the code you want to retrieve. You can set several expected codes, separate by coma. That's useful, if it doesn't matter if a request was successfully.

I.e.: You want to delete a file. But it's not sure that the file exists.

If the file exists, server reply is: 250 DELE command successful.

If it doesn't exist, reply is: 550 file does not exists.

So you have to set the expected code to "250,550;" and blError is in both causes FALSE.

Semicolon marks the end of expected answer codes.

### Reply types

Code	Significance
110	Restart marker reply. In this case, the text is exact and not left to the particular implementation; it must read: MARK yyyy = mmmm Where yyyy is User-process data stream marker, and mmmm server's equivalent marker (note the spaces between markers and "=").
120	Service ready in nnn minutes.
125	Data connection already open; transfer starting.
150	File status okay; about to open data connection.
200	Command okay.
202	Command not implemented, superfluous at this site.
211	System status, or system help reply.
212	Directory status.
213	File status.
214	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.
215	NAME system type. Where NAME is an official system name from the list in the Assigned Numbers document.
220	Service ready for new user.
221	Service closing control connection. Logged out if appropriate.
225	Data connection open; no transfer in progress.

226	Closing data connection. Requested file action successful (for example, file transfer or file abort).
227	Entering Passive Mode (h1,h2,h3,h4,p1,p2).
230	User logged in, proceed.
250	Requested file action okay, completed.
257	"PATHNAME" created.
331	User name okay, need password.
332	Need account for login.
350	Requested file action pending further information.
421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
425	Can't open data connection.
426	Connection closed; transfer aborted.
450	Requested file action not taken. File unavailable (e.g., file busy).
451	Requested action aborted: local error in processing.
425	Requested action not taken. Insufficient storage space in system.
500	Syntax error, command unrecognized. This may include errors such as command line too long.
501	Syntax error in parameters or arguments.
502	Command not implemented.
503	Bad sequence of commands.
504	Command not implemented for that parameter.
530	Not logged in.
532	Need account for storing files.
550	Requested action not taken. File unavailable (e.g., file not found, no access).
551	Requested action aborted: page type unknown.
552	Requested file action aborted. Exceeded storage allocation (for current directory or dataset).
553	Requested action not taken. File name not allowed.

---

## Data transfer

### Simple download

Now you are ready to download a file. Prepare and login like before. Than call the following subroutine.

```
call ftpGetFile(slGetFilePath$, slGetFileContent$, slActiveIpPort$, &
                EXPECTED_CODE_GET_FILE, blError)
```

`slGetFilePath$` contains the path of the file you want to download.

In `slGetFileContent$` the content of the file will be saved.

`slActiveIpPort$` will be explained later. Let it empty.

`EXPECTED_CODE_GET_FILE` is the server code you expect. For successful data transfer its "226;"

`blError` is `FALSE` if the server returns your expected code.

At last, quit the ftp session and save the ftp log data in flash.

### Total code

```
`#define SET_LOCAL_DATA_PORT
`#define SET_LOCAL_COMM_PORT

#define PRINT_STATUS_MSG
#define DISPLAY_DEV_NUM
```

```

#define MAKE_FTP_LOG

#define RECV_DATA_IN_LOG
#define SENT_DATA_IN_LOG

#include define_a.inc
#include ufunc3.inc
#include ftp_needed_files.inc

Task Main
    BYTE blError
    STRING slGetFilePath$, slGetFileContent$(30k)

    'Install graphic display driver with device number DISPLAY_DEV_NUM
    'Install serial device driver.

    slGetFilePath$ = "file_to_download.txt"

    call prepareClient(blError)
    if blError = FALSE then
        call ftpLogin(FTP_IP_ADDRESS, FTP_COMMAND_PORT, FTP_USERNAME, &
                     FTP_PASSWORD, blError)

        if blError = FALSE then
            call ftpGetFile(slGetFilePath$, slGetFileContent$, "", &
                           EXPECTED_CODE_GET_FILE, blError)
            call ftpQuitSession(blError)
        endif
    endif

    call putIntoFlash(sFtpLog$)
End

```

## Display output

If you didn't change the "*LANG\_ENG.INC*", the display shows:

```

Status: ☺ successfully executed – ftpLogin()
Status: ☺ successfully executed – ftpGetFile()
Status: ☺ successfully executed – ftpQuitSession()

```

## Ftp Log

The flash has the following entries:

The server responses are **blue**, the client requests are **red** and retrieved data is **green**.  
220 FTP Test Server ready.

**USER ftpUser**

331 Password required for ftpUser.

**PASS ftpPass**

230 User ftpUser logged in.

**PASV**

227 Entering Passive Mode (192,168,1,42,6,105)

**RETR download.txt**

150 Data connection accepted from 192.168.1.46:28966; transfer starting for download.txt (748 bytes).

226 Transfer ok

Blindtext im Kopf

Achtung! Dieser Blindtext wird gerade durch 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen



Erregungszustand versetzt, der sich über den Sehnerv in dem hinteren Teil Ihres Gehirns ausbreitet. Von dort aus überträgt sich die Erregung in Sekundenbruchteilen auch in andere Bereiche Ihres Großhirns. Ihr Stirnlappen wird stimuliert. Von dort aus gehen jetzt Willensimpulse aus, die Ihr zentrales Nervensystem in konkrete Handlungen umsetzt. Kopf und Augen reagieren bereits. Sie folgen dem Text, nehmen die darin enthaltenen Informationen auf wie ein Schwamm. Nicht auszudenken, was mit Ihnen hätte passieren können, wenn dieser Blindtext durch einen echten Text ersetzt worden wäre.

QUIT

221 Bye bye ...

## Simple upload

Now we want to download a file and save it as new file on the server with the following subroutine:

```
call ftpStoreFile(slStoreFilePath$, slStoreFileContent$, &
                  slActiveIpPort$, EXPECTED_CODE_GET_FILE, blError)
```

slStoreFilePath\$ contains the path of the file you want to store on server.  
slStoreFileContent\$ is the content of the new uploaded file. Will be the same as slGetFileContent\$, in this example.

At last, quit the ftp session and save the ftp log data in flash.

### Total code

```
`#define SET_LOCAL_DATA_PORT
`#define SET_LOCAL_COMM_PORT

#define DISPLAY_DEV_NUM #LCD
#define PRINT_STATUS_MSG

#define MAKE_FTP_LOG

`#define RECV_DATA_IN_LOG
#define SENT_DATA_IN_LOG

#include define_a.inc
#include ufunc3.inc
#include ftp_needed_files.inc
```

#### Task Main

```
BYTE blError
STRING slGetFilePath$, slGetFileContent$(30k), &
      slStoreFilePath$, slStoreFileContent$(30k)

`Install graphic display driver with device number DISPLAY_DEV_NUM
`Install serial device driver.

call prepareClient(blError)
if blError = FALSE then
    call ftpLogIn(FTP_IP_ADDRESS, FTP_COMMAND_PORT, FTP_USERNAME, &
                  FTP_PASSWORD, blError)

    if blError = FALSE then
        slGetFilePath$ = "file_to_download.txt"
        call ftpGetFile(slGetFilePath$, slGetFileContent$, "", &
                        EXPECTED_CODE_GET_FILE, blError)
        if blError = FALSE then
            slStoreFilePath$ = "[copy]" + slGetFilePath$
            slStoreFileContent$ = slGetFileContent$
```

```

        call ftpStoreFile(slStoreFilePath$, &
                           slStoreFileContent$, &
                           "", EXPECTED_CODE_GET_FILE, &
                           blError)
    endif
    call ftpQuitSession(blError)
endif
endif

call putIntoFlash(sFtpLog$)
End

```

## Display output

If you didn't change the "LANG\_ENG.INC", the display shows:

```

Status: ☺ successfully executed – ftpLogin()
Status: ☺ successfully executed – ftpGetFile()
Status: ☺ successfully executed – ftpStoreFile()
Status: ☺ successfully executed – ftpQuitSession()

```

## Ftp Log

The flash has the following entries:

The server responses are **blue**, the client requests are **red** and sent data is **green**.  
 220 FTP Test Server ready.

**USER ftpUser**

331 Password required for ftpUser.

**PASS ftpPass**

230 User ftpUser logged in.

**PASV**

227 Entering Passive Mode (192,168,1,42,60,113)

**RETR download.txt**

150 Data connection accepted from 192.168.1.46:29014; transfer starting for download.txt (748 bytes).

226 Transfer ok

**PASV**

227 Entering Passive Mode (192,168,1,42,240,198)

**STOR [copy]download.txt**

150 Data connection accepted from 192.168.1.46:29015; transfer starting for [copy]download.txt.

**Blindtext im Kopf**

Achtung! Dieser Blindtext wird gerade durch 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen Erregungszustand versetzt, der sich über den Sehnerv in dem hinteren Teil Ihres Gehirns ausbreitet. Von dort aus überträgt sich die Erregung in Sekundenbruchteilen auch in andere Bereiche Ihres Großhirns. Ihr Stirnlappen wird stimuliert. Von dort aus gehen jetzt Willensimpulse aus, die Ihr zentrales Nervensystem in konkrete Handlungen umsetzt. Kopf und Augen reagieren bereits. Sie folgen dem Text, nehmen die darin enthaltenen Informationen auf wie ein Schwamm. Nicht auszudenken, was mit Ihnen hätte passieren können, wenn dieser Blindtext durch einen echten Text ersetzt worden wäre.

226 File received ok.

QUIT  
221 Bye bye ...

## Get file list

This routine causes a list to be sent from the server to the passive DTP. If the pathname specifies a directory or other group of files, the server should transfer a list of files in the specified directory. If the pathname specifies a file then the server should send current information on the file. A null argument implies the user's current working or default directory. The data transfer is over the data connection in type ASCII or type EBCDIC. (The user must ensure that the TYPE is appropriately ASCII or EBCDIC). Since the information on a file may vary widely from system to system, this information may be hard to use automatically in a program, but may be quite useful to a human user.

```
call ftpListDir(slListDirPath$, slListDirContent$, slLocalIpPort$, &  
                EXPECTED_CODE_GET_FILE, blError)
```

slListDirPath\$ is the path you want to get the file list from.  
slListDirContent\$ gets the file list.

At last, quit the ftp session and save the ftp log data in flash.

## Total code

```
`#define SET_LOCAL_DATA_PORT  
`#define SET_LOCAL_COMM_PORT  
  
#define DISPLAY_DEV_NUM #LCD  
#define PRINT_STATUS_MSG  
  
#define MAKE_FTP_LOG  
  
#define RECV_DATA_IN_LOG  
`#define SENT_DATA_IN_LOG  
  
#include define_a.inc  
#include ufunc3.inc  
#include ftp_needed_files.inc
```

### Task Main

```
    BYTE blError  
    STRING slListDirPath$, slListDirContent$(30k)  
  
    `Install graphic display driver with device number DISPLAY_DEV_NUM  
    `Install serial device driver.  
  
    call prepareClient(blError)  
    if blError = FALSE then  
        call ftpLogin(FTP_IP_ADDRESS, FTP_COMMAND_PORT, FTP_USERNAME, &  
                    FTP_PASSWORD, blError)  
        if blError = FALSE then  
            slListDirPath$ = "directory/"  
            call ftpListDir(slListDirPath$, slListDirContent$, "", &  
                        EXPECTED_CODE_LIST_DIR, blError)  
            call ftpQuitSession(blError)  
        endif  
    endif  
  
    call putIntoFlash(sFtpLog$)
```

**End**

## Display output

If you didn't change the "LANG\_ENG.INC", the display shows:

Status: ☺ successfully executed – ftpLogin()

Status: ☺ successfully executed – ftpListDir()  
Status: ☺ successfully executed – ftpQuitSession()

### Ftp Log

The flash has the following entries:

The server responses are **blue**, the client requests are **red** and sent data is **green**.

220 FTP Test Server ready

**USER ftpUser**

331 Password required for ftpUser.

**PASS ftpPass**

230 User ftpUser logged in.

**PASV**

227 Entering Passive Mode (192,168,1,42,78,168)

**LIST**

150 Data connection accepted from 192.168.1.46:29017; transfer starting.

226 Transfer ok

```
drwxr-xr-x  1 ftp  ftp      0   Aug 17  10:26 renamed dir
-rw-r--r--  1 ftp  ftp    748  Aug 23  13:45 [copy]download.txt
-rw-r--r--  1 ftp  ftp   1704  Jul 29   16:35 arbeit.html
-rw-r--r--  1 ftp  ftp   3114  Dec 17  2004 die frau.html
-rw-r--r--  1 ftp  ftp    748  Aug 19  16:29 download.txt
-rw-r--r--  1 ftp  ftp   1763  Aug 17  10:11 filelist.txt
-rw-r--r--  1 ftp  ftp    439  Aug 17  16:12 help_site.txt
-rw-r--r--  1 ftp  ftp    279  Aug 17  16:12 neue_datei.txt
-rw-r--r--  1 ftp  ftp     13  Aug 17  16:12 zone.txt
```

**QUIT**

221 Bye bye ...

## Append data

If you want to append data in an existing file, call the following subroutine:

```
call ftpAppendData(slAppendFilePath$, slAppendFileContent$,
                    slLocalIpPort$, ECXPECTED_CODE, blError)
```

slAppendFilePath\$ is the file you want to append data string from  
slAppendFileContent\$.

### Total code

```
`#define SET_LOCAL_DATA_PORT
`#define SET_LOCAL_COMM_PORT

#define DISPLAY_DEV_NUM #LCD
#define PRINT_STATUS_MSG

#define MAKE_FTP_LOG

`#define RECV_DATA_IN_LOG
#define SENT_DATA_IN_LOG

#include define_a.inc
#include ufunc3.inc
#include ftp_needed_files.inc
```

**Task** Main

**BYTE** blError

**STRING** slAppendFilePath\$, slAppendFileContent\$(30k)

```

`Install graphic display driver with device number DISPLAY_DEV_NUM
`Install serial device driver.

call prepareClient(blError)
if blError = FALSE then
    call ftpLogin(FTP_IP_ADDRESS, FTP_COMMAND_PORT, FTP_USERNAME, &
        FTP_PASSWORD, blError)
    if blError = FALSE then
        slAppendFilePath$ = "file.end"
        slAppendFilePath$ = "this string will be append to " + &
            slAppendFilePath$
        call ftpAppendData(slAppendFilePath$, &
            slAppendFileContent$, "", &
            EXPECTED_APPE_FILE, blError)
        call ftpQuitSession(blError)
    endif
endif

call putIntoFlash(sFtpLog$)
End

```

### Display output

If you didn't change the "*LANG\_ENG.INC*", the display shows:

```

Status: ☺ successfully executed – ftpLogin()
Status: ☺ successfully executed – ftpAppendData()
Status: ☺ successfully executed – ftpQuitSession()

```

### Ftp Log

The flash has the following entries:

The server responses are **blue**, the client requests are **red** and sent data is **green**.

220 FTP Test Server

**USER ftpUser**

331 Password required for ftpUser.

**PASS ftpPass**

230 User ftpUser logged in.

**PASV**

227 Entering Passive Mode (192,168,1,42,32,91)

**APPE file.end**

150 APPE supported. Ready to append file "file.end" at offset 0. (192.168.1.46:29019)

this string will be append to the file file.end

226 File received ok.

**QUIT**

221 Bye bye ...

### Data transfer in active mode

All subroutines so far, transfer the data in passive mode. That means, the client requests for an unprivileged port. The server sends a response with the port and the client build up the connection for the data transfer.

In active mode, the client sends a port command, with port number, where it listens for data or sends data. The server build up the connection and the data can be transferred.

If you want to transfer data in active mode, you have to set the parameter `slLocalIpPort$` like this:

```
slLocalIpPort$ = "ip1,ip2,ip3,ip4,p1,p2"
```

I.e.: Your local port is 10000 and ip 192.168.1.2, so you have to set `slLocalIpPort$ = p1 x 256 + p2 = 10000 --> "192,168,1,2,39,17"`

Now we pick up the first download sample and will download the file in active mode with the previous settings.

### Total code

```
`#define SET_LOCAL_DATA_PORT
`#define SET_LOCAL_COMM_PORT

#define PRINT_STATUS_MSG
#define DISPLAY_DEV_NUM

#define MAKE_FTP_LOG

#define RECV_DATA_IN_LOG
`#define SENT_DATA_IN_LOG

#include define_a.inc
#include ufunc3.inc
#include ftp_needed_files.inc

Task Main
    BYTE blError
    STRING slGetFilePath$, slGetFileContent$(30k)

    `Install graphic display driver with device number DISPLAY_DEV_NUM
    `Install serial device driver.

    slGetFilePath$ = "file_to_download.txt"

    call prepareClient(blError)
    if blError = FALSE then
        call ftpLogin(FTP_IP_ADDRESS, FTP_COMMAND_PORT, FTP_USERNAME, &
            FTP_PASSWORD, blError)

        if blError = FALSE then
            call ftpGetFile(slGetFilePath$, slGetFileContent$, &
                "192,168,1,2,39,17", EXPECTED_CODE_GET_FILE, &
                blError)
            call ftpQuitSession(blError)
        endif
    endif

    call putIntoFlash(sFtpLog$)
End
```

### Display output

If you didn't change the "*LANG\_ENG.INC*", the display shows:

```
Status: ☺ successfully executed – ftpLogin()
Status: ☺ successfully executed – ftpGetFile()
Status: ☺ successfully executed – ftpQuitSession()
```

### Ftp Log

The flash has the following entries:

The server responses are blue, the client requests are red and retrieved data is green.  
220 FTP Test Server ready.

USER ftpUser

331 Password required for ftpUser.

PASS ftpPass

230 User ftpUser logged in.

PORT 192,168,1,46,39,17

200 Port command successful.

RETR download.txt

150 Opening data connection for download.txt (748 bytes).

226 Transfer ok

Blindtext im Kopf

Achtung! Dieser Blindtext wird gerade durch 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen Erregungszustand versetzt, der sich über den Sehnerv in dem hinteren Teil Ihres Gehirns ausbreitet. Von dort aus überträgt sich die Erregung in Sekundenbruchteilen auch in andere Bereiche Ihres Großhirns. Ihr Stirnlappen wird stimuliert. Von dort aus gehen jetzt Willensimpulse aus, die Ihr zentrales Nervensystem in konkrete Handlungen umsetzt. Kopf und Augen reagieren bereits. Sie folgen dem Text, nehmen die darin enthaltenen Informationen auf wie ein Schwamm. Nicht auszudenken, was mit Ihnen hätte passieren können, wenn dieser Blindtext durch einen echten Text ersetzt worden wäre.

QUIT

221 Bye bye ...

## Restart

The argument field represents the server marker at which file transfer is to be restarted. This command does not cause file transfer but skips over the file to the specified data checkpoint. This command shall be immediately followed by the appropriate FTP service command which shall cause file transfer to resume.

```
slRestFilePos$ = "0"  
call ftpRestart(slRestFilePos$, EXPECTED_CODE_RESTART, blError)
```

The reply is type 350.

## Representation type

The argument specifies the representation type as described in the Section on Data Representation and Storage. Several types take a second parameter. The first parameter is denoted by a single Telnet character, as is the second Format parameter for ASCII and EBCDIC; the second parameter for local byte is a decimal integer to indicate Byte size. The parameters are separated by a <SP> (Space, ASCII code 32). The following codes are assigned for type:

A - ASCII	\	/	N - Non-print
E - EBCDIC			T - Telnet format effectors
	-><-		C - Carriage Control (ASA)
	/	\	
I - Image			
L <byte size>			- Local byte Byte size

Use the following subroutine to set the type.

```
slReprTypeParam$ = "I"
```

```
call ftpReprType(slReprTypeParam$, EXPECTED_CODE_TYPE, blError)
```

The server returned code is usually 200.

## Transfer mode

The argument is a single Telnet character code specifying the data transfer modes described in the Section on Transmission Modes.

The following codes are assigned for transfer modes:

- S - Stream
- B - Block
- C - Compressed

The default transfer mode is Stream.

You can set the mode with this subroutine:

```
slTransferModeParam$ = "S"  
call ftpTransferMode(slTransferModeParam$, EXPECTED_CODE_TMODE, blError)
```

The server returned code is usually 200.

## File structure

The argument is a single Telnet character code specifying file structure described in the Section on Data Representation and Storage.

The following codes are assigned for structure:

- F - File (no record structure)
- R - Record structure
- P - Page structure

The default structure is File.

Set the mode with this subroutine:

```
slFileStrucParam$ = "F"  
call ftpFileStructure(slFileStrucParam$, EXPECTED_CODE_STRUCTURE, blError)
```

The server returned code is usually 200.

---

## File and directory operations

### Delete file

This routine causes the file specified in the pathname to be deleted at the server site. If an extra level of protection is desired (such as the query, "Do you really wish to delete?"), it should be provided by the user-FTP process.

```
slDeleteFilePath$ = "FileToDelete.end"  
call ftpDeleteFile(slDeleteFilePath$, EXPECTED_CODE_DELETE_FILE, blError)
```

Server returned code is usually 200

### File Size

This routine needs one argument: the path of the file you want to get the size from. The file size will be saved in `llFileSize`.



```
slFileSizePath$ = "file.end"  
call ftpFileSize(slFileSizePath$, llFileSize, EXPECTED_CODE_SIZE, blError)
```

The server returned code is usually 213.

## Make new directory

This routine causes the directory specified in the pathname to be created as a directory (if the pathname is absolute) or as a subdirectory of the current working directory (if the pathname is relative).

```
slDirectoryName$ = "new/directory"  
call ftpMakeDir(slDirectoryName$, EXPECTED_CODE_MAKE_DIR, blError)
```

The server returned code is usually 257.

## Delete directory

This routine causes the directory specified in the pathname to be removed as a directory (if the pathname is absolute) or as a subdirectory of the current working directory (if the pathname is relative).

```
slDirectoryName$ = "directory/to/delete"  
call ftpDeleteDir(slDirectoryName$, EXPECTED_CODE_DELETE_DIR, blError)
```

The server returned code is usually 250.

## Rename file or directory

This routine renames a file or empty directory. Two arguments must be given: First one specifies the old pathname of the file which is to be renamed. The second one specifies the new pathname.

```
slRenamePath$ = "rename/directory"  
slNewName$ = "rename/new name"  
call ftpRename(slRenamePath$, slNewName$, EXPECTED_CODE_RENAME, blError)
```

The server returned code is usually 350.

## Change working directory

This routine allows the user to work with a different directory or dataset for file storage or retrieval without altering his login or accounting information. Transfer parameters are similarly unchanged. The argument is a pathname specifying a directory or other system dependent file group designator.

```
slChangeDir$ = "change/to/directory/"  
call ftpChangeDir(slChangeDir$, EXPECTED_CODE_CHANGE_DIR, blError)
```

The server returned code is usually 250.

## Change to parent directory

This routine is a special case of CWD, and is included to simplify the implementation of programs for transferring directory trees between operating systems having different syntaxes for naming the parent directory. The reply codes shall be identical to the reply codes of CWD.

```
call ftpParentDir(EXPECTED_CODE_PARENT_DIR, blError)
```

The server returned code is usually 250.

## Get working directory

This command causes the name of the current working directory to be returned in the reply and stored in `slWorkingDir$`. It takes no arguments and forces the server to return a 257 message with the directory path listed as an argument.

257 "ftp\_projekt/" is current directory

So you can separate the directory name between the quotes.

```
call ftpWorkingDir(slWorkingDir$, EXPECTED_CODE_WORKING_DIR, blError)
```

The server returned code is usually 257.

---

## User rights

This command is used by the server to provide services specific to his system that are essential to file transfer but not sufficiently universal to be included as commands in the protocol. The nature of these services and the specification of their syntax can be stated in a reply to the help routine.

### Arguments:

**chmod [option] <chmod#> <dir/file.end>**: sets or changes the user rights of files or directories. Can only used by owner or system admin.

#### Label:

Each type (Owner/Group/Other) needs a number. So the chmod command contains 3 numbers.

read	write	execute	chmod
=====	=====	=====	=====
r	w	x	7
r	w	-	6
r	-	x	5
r	-	-	4
-	w	x	3
-	w	-	2
-	-	x	1
-	-	-	0

**chgrp [option] <group name> <dir/file.end>** : changes the group of a file or directory. You can get a group list with the argument GROUPS (if supported).

**chown [option] <owner name> <dir/file.end>** : Changes the owner of a file.

There are different arguments supported by different servers. For a list with supported arguments, use the *ftpHelp()* subroutine with the argument parameter *SITE*.

If you want to change the user rights for *file.end* so that the owner can write, read and execute it, the group may read and execute and all other haven't any rights, the chmod argument is:

```
chmod 750 file.end
```

Set the argument and call the following subroutine:

```
slChmod$ = "chmod 750 file.end"
call ftpChParam(slChmod$, slChmodContent$, EXPECTED_CODE_CHMOD, blError)
```

`slChmodContent$` will be similarly:  
`200 CHMOD command successful.`

---

## Other routines

### Help

This routine shall cause the server to send helpful information regarding its implementation status over the control connection to the user. The command may take an argument (e.g., any command name) and return more specific information as a response. It is suggested that HELP be allowed before entering a USER command. The server may use this reply to specify site-dependent parameters, e.g., in response to HELP SITE.

```
slHelpArgument$ = "SITE"  
call ftpHelp(slHelpArgument$, slHelpContent$, EXPECTED_CODE_HELP, blError)
```

The reply is type 211 or 214.

### No operation

This command does not affect any parameters or previously entered commands. It specifies no action other than that the server send an OK reply.

```
call ftpEmptyRequest(EXPECTED_CODE_EMPTY_REQ, blError)
```

The server returned code is usually 200.