WAR FTP DAEMON

Ver. 1.65. Released at Apr 24. 1997

WAR FTP Daemon is the premier FTP server for Windows 95 and NT. No other FTP server application brings together the UNIX style security features, a BBS like Windows interface for the system operator, the extreme flexibility, Win95 and NT 4 OLE support, the multithreading design and the advanced software technology and performance provided by this package!

And despite the technical quality, the package is released as freeware; available on CD's, and spread all over Internet on a large number of HTTP servers, FTP servers and BBS systems. Provided that you accept the license agreement, you can use this software freely without registering or paying for it.

War FTP Daemon follows the RFC 959 and 1123 specifications (with industry standard extensions) of the FTP protocol. It is running as an user application, or (optionally) as a system service on NT. On secure systems, the users that log into the system will have access to the files available for the (operating system) user-account that started the program. Within this limitation, it also provides a powerful internal directory level security subsystem.

War FTP Daemon is written for personal and professional use. It’s popularity is growing very fast, and already after less than 6 months availability, the server was established as the premier FTP server for the 32 bit Windows platform, with top ratings on all major software listings that have evaluated/tested the program.

War FTP Daemon should not be used for any sensitive files, as anyone with a TCP/IP protocol analyzer and access to the physical transmission layer (cables) will be able to tap any data transmitted. (This is not a limitation in this program, it is a general limitation with the FTP protocol specification and applies to all servers using the standard FTP protocol interface.) Be aware that most of the common used encryption programs and protocols can be easily cracked by hackers, industrial “spy’s” and law enforcement agencies.

Table of contents

The latest version of the server is always available at:

http://www.jgaa.com
http://home.sol.no/jgaa/

The web site also host a FAQ and updated support and contact information.

You are welcome to use the newsgroup news:alt.comp.jgaa for comments, suggestions, questions and to stay updated on the development of new versions.

Note: The War FTP Daemon is released as copyrighted freeware. See the **license** agreement for details. You do not need to register or pay in order to use this software.

Author is:
Jarle Aase, Bergen, Norway.
For information: info@jgaa.com (email, auto-responder)
Private Email: jgaa@jgaa.com

The design of the War FTP Daemon

Users, groups and classes

A user in the FTP Daemon is a record of information including a login-name, a password, and a long list of different security properties. To ease the maintenance of the system, each user belong in a user-group and a user-class. If a security option is not set explicit on a user, the daemon looks in the group, class and finally, - if it still can't find a determinant value, in the system default setup.

- ***NOTE**: A group or a class can NOT log in to the daemon. **Only*** users *can* ***log*** *in.*

Let's take an example. A user has logged in and issues a "`CD /pub/windows`" command from his FTP client program. The server looks in the users_table of paths without finding the path. It goes on to the group, fails, goes on to the class, where (in this case) the path was defined. However, there was not defined any permissions. The server then look up the users_default file permissions, where it fails to find any default read or deny-read flags. It goes on to the group, class and finally the default setup, where it finds a read flag. The user is then given access to the file. If the daemon had encountered a deny-read flag anywhere in this process, the user would have been denied access.

The advantage of this design is that it makes it easy to maintain a large system, with many users and even more directories. When you attach a user to a group, it inherits all the permissions of this group. When you attach a user to a class, it inherits all the permissions of the class.

The disadvantage is that it can be difficult to keep track of the permissions in all 4 layers

Default
Class
Group
User

To resolve this problem, the server has a number of reports that will show you the actual permissions of a user.

If you don't need all this flexibility, you can simply turn it off and maintain only the user and default properties. If you use the system on your home-PC, providing a FTP service to your friends, I will suggest this. You still have the same level of security over the files you **don't** want to share.

The advanced options are turned off by default. To enable them, go to the menu General Options tab.

All security functions search in this order:

User > Group > Class > Default.

If a determinant value is found, it stops the search and uses that value. This means that if a class is denied access, while one of the users in this class has an explicit Account-open flag, this user will get access, while the rest of the users in his class will be denied access.

When you examine the security dialogs, you will find that most options has three states:

☑ Yes
□ No
☑ Default

Default means: Look at the next level.

Root dir and home dir

This FTP server introduces the concept of both a home and a root directory. On other servers for DOS/Windows, this is usually not available. However, on UNIX/NT FTP servers, the users have a regular user account with the operating system, and hence a home directory and a root directory. This server does not touch the NT user database, but keeps it's own user database where all restrictions and permissions are kept. This database is encrypted.

The root directory is the highest level of the directory structure the user is allowed to move. Usually you will "map" any allowed paths outside the root directory, making it look like the path actually are a part of the users root path.

Example:

C:\usr\pub\ftp	> Root directory	
D:\Apps	> Mapped to root	> (looks like) /c/usr/pub/ftp/Apps
		> (or, if root maps to root) /Apps

In addition to this, each user can have a "home" directory. This is the directory that will be the current directory when the user logs on. The server does not give any special permissions to the home directory.

Using secure home dirs

Some FTP configurations will offer users their own private home directories. You can do this by locating the home directories in a path where the users normally don't have access:

level	Path	Flags
Group	C:\usr\pub\ftp	Root directory
Group	C:\usr\pub\ftp\users	Directory specified with deny flag, preventing any access.
User	C:\usr\pub\ftp\users\JoeD	Home dir of JoeD, mapped to root

This scenario denies everyone access to the users dir, except JoeD that have full permissions to his own directory, and will find the dir, not under users, but under /c/pub/ftp/JoeD or just /JoeD (if root is mapped to root). However, JoeD will only have access to his own homedir, not to any other user-directories in the users directory structure.

See the File Access section for more information on how to set up directories and permissions.

System security

When the server is first started, it adds it's own working directory to the system wide list of paths, with the deny flag set. This prevents any FTP access to the FTP directory.

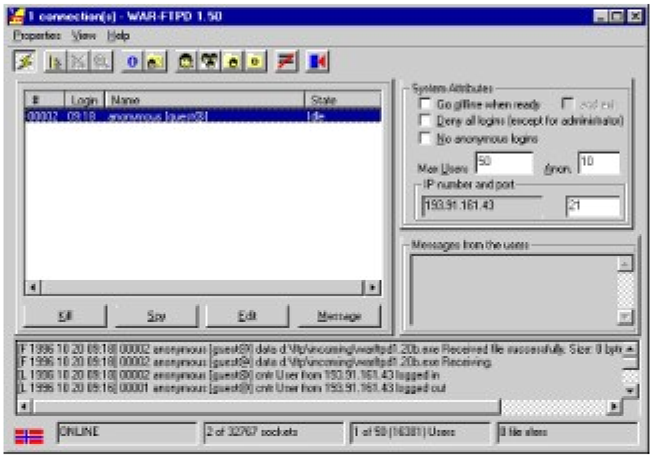
The database "FtpDaemon.dat" is encrypted. This makes it useless for any browsing. User passwords can not be seen unless you monitor the TCP/IP packages, or turn on the logging of the control connection.

License agreement

The WAR FTP Daemon is a PC based FTP server for Win95 and Windows NT 3.51 or better. It is released as “freeware” under the following conditions:

- You might not charge anything for the distribution of this software, but the actual costs of media, shipping etc.
- This software comes without any guarantee of any kind. The author TAKE NO RESPONSIBILITY FOR ANY DAMAGE, LOSS OF INCOME, OR ANY OTHER PROBLEMS YOU MIGHT EXPERIENCE FROM USING THIS SOFTWARE.
- This software is NOT TO BE USED by any governmental or mainstream political institutions in any country. The freeware policy does ONLY apply for private individuals and private corporations. The governmental and mainstream political organizations can go and get their software somewhere else. Universities, schools and other public educational institutions can use this software as they wish.
- Jarle Aase keeps the full Copyright of all this software and documentation, and reserves the right to change this policy at any time.

The system console (Main window)



#	Login	Name	State
00002	09:18	anonymous [guest@]	Idle

System Attributes

- □ Go offline when ready □ and exit
- □ Deny all logins (except for administrator)
- □ No anonymous logins

Max Users 50 Anon. 10

IP number and port: 193.91.161.43 21

```
[F 1996 10 20 09:18] 00002 anonymous [guest@] data d:\ftp\incoming\warftpd1.20b.exe Received file successfully. Size: 0 byt
[F 1996 10 20 09:18] 00002 anonymous [guest@] data d:\ftp\incoming\warftpd1.20b.exe Receiving
[L 1996 10 20 09:18] 00002 anonymous [guest@] cntr User from 193.91.161.43 logged in
[L 1996 10 20 09:16] 00001 anonymous [guest@] cntr User from 193.91.161.43 logged out
```

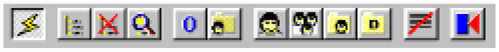
ONLINE | 2 of 32767 sockets | 1 of 50 (16381) Users | 0 file xfers

The system console is the main window for the War FTP Daemon.

The console contains the following parts:

- Menu bar
- Toolbar
- User List
- System Attributes
- Messages from the Users
- Log Window
- System Status

Toolbar



Go Online
Go Offline
Retart the Virtual File System
Stop the Virtual File System
View the Virtual File System
Options
Security: Edit All
Security: Edit User
Security: Edit Group
Security: Edit Class
Security: Edit Default setup
Clear Log Window
Exit

Menu

- Properties
- View
- Help

Help

- Contents
- Search For Help On
- Bug Report
- About

View

- Reports
- Messages
- Macros
- Clear Log Window

Reports

- User Access Privileges
- User Home and Root Dir
- User Paths
- Password Properties
- Up Download Statistics

Properties Menu

General

Security

Options System configuration

VfSys

Stop Service

Start Service

Import

Exit

Security menu items

Edit User	Edit the security properties on the user level
Edit Group	Edit the security properties on the group level
Edit Class	Edit the security properties on the class level
Edit Default	Edit the security properties on the default level
Edit All	Edit the security properties on all levels

Note. This help file does only explain the Edit All mode, as the dialog used for all levels are very similar.

Virtual File System menu items

- Restart
- Stop
- View
- Flush

See also: Virtual File System. and Virtual File System Options

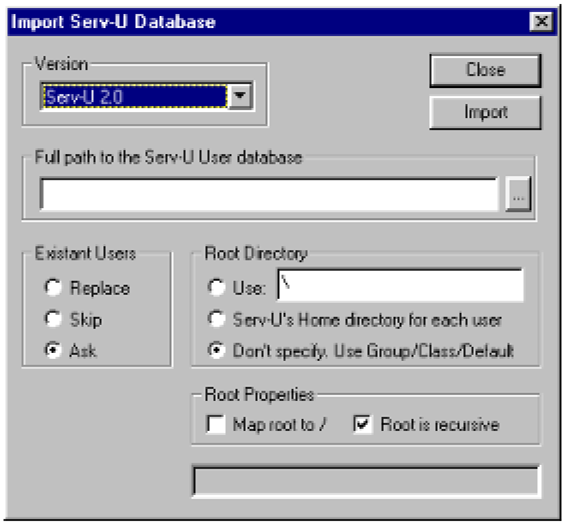
Import Menu Items

Serv-U Database

The War FTP Daemon is capable of importing the user database from other FTP servers. This feature is added to ease migration for sites with large number of users.

Note: Every FTP server has it's own design and it's own way of doing things. There is no standard data exchange format. This gives a number of limitations in the import process. The import module will try to make the migration as smooth as possible, but you should check the different reports to make sure that everything is OK. You should also log in as different users and verify that the permissions are correct. (If you have 500 users, you don't have to log in 500 times).

Import a Serv-U Database



Version: Serv-U 2.0

Full path to the Serv-U User database:

Existant Users
○ Replace
○ Skip
● Ask

Root Directory
○ Use: \
○ Serv-U's Home directory for each user
● Don't specify. Use Group/Class/Default

Root Properties
□ Map root to /
☑ Root is recursive

Serv-U stores it's user information in a (almost) standard Windows .ini file named **Serv-u.ini**. All information is available as clear text, except the password, that is scrambled with the DES13 method. War FTP Daemon has it's own way of handling passwords, but it can also deal with DES13. The imported users will also keep their old passwords unchanged.

Version

Select the version of the database you will import.

Existent Users

Replace. If a user in the Serv-U database has the same name as an existent War user, the current War user will be deleted and replaces with the user from Serv-U

Skip. If a user in the Serv-U database has the same name as an existent War user, the user will not be imported and the current War user will remain untouched.

Ask. The server will ask what to do.

Note: The users Anonymous and ALL will not be imported.

Root Directory

Serv-U does not have root path's. War therefore have to know how you want it to handle the imported user accounts.

Use *Path*. Specify a path that will be assigned as root for all the imported users.

Serv-U's Home... Set the users home path in the Serv-U database as root path for each user. *This will normally be a bad idea.*

Don't... War will leave the users without root-path's so that the root path can be defined at the Class or Default level.

Root Properties

If you have chosen to give the users root-path's, you can specify the basic root properties

Map... The users root path will appear as / (top level) to the user.

Recursive... The permissions granted for root (which is the default permissions) will be given recursively to all directories beneath the root. This property should always be enabled.

Clear Log Window

Toolbar:

Menu: Properties/Vfsys

Clears the log window on the System Console.

The log file is not affected.

Stop the Virtual File System

Toolbar: ☒
Menu: Properties/Vfsys

Stops the Virtual File System.

This can be done while the server is online, but not if there are any users logged in.

If the Virtual File System is stopped, the server will fall back to the normal file system. Users with \ as root path will get access to all drives and paths, except paths that are defined with the deny flag.

See also: Virtual File System. and Virtual File System Options

Restart the Virtual File System

Toolbar:
Menu: Properties/Vfsys

Starts or re-starts the Virtual File System. The Virtual File System can be started when the server is offline and when it is online and no one is logged in. If it already is running it can be restarted at any time.

When the Virtual File System is started (or restarted) it will scan all directories in the Virtual File System path and create an internal list of all directories and files. This list will be modified if users make changes to the file system, but not if files are created, deleted or moved from Windows (or DOS). If you make changes to the directories, make sure to restart the Virtual File System so that the user gets the directory listings correct.

Note that all users share the same internal file list. If a user logs off and the on again, the Virtual File system will not change. Also, taking the server offline and online again does not restart the Virtual file System. When it is running you must manually execute this command if you want to refresh the internal lists.

If a user is accessing the Virtual File System (i.e. creating a .SysIdx.txt file), when you restart it, a new list will be built for all future access. The old list will be trashed when the user end's the current operation. (In this example - the next directory listing or .SysIdx.txt file will use the newly created Virtual File System list.).

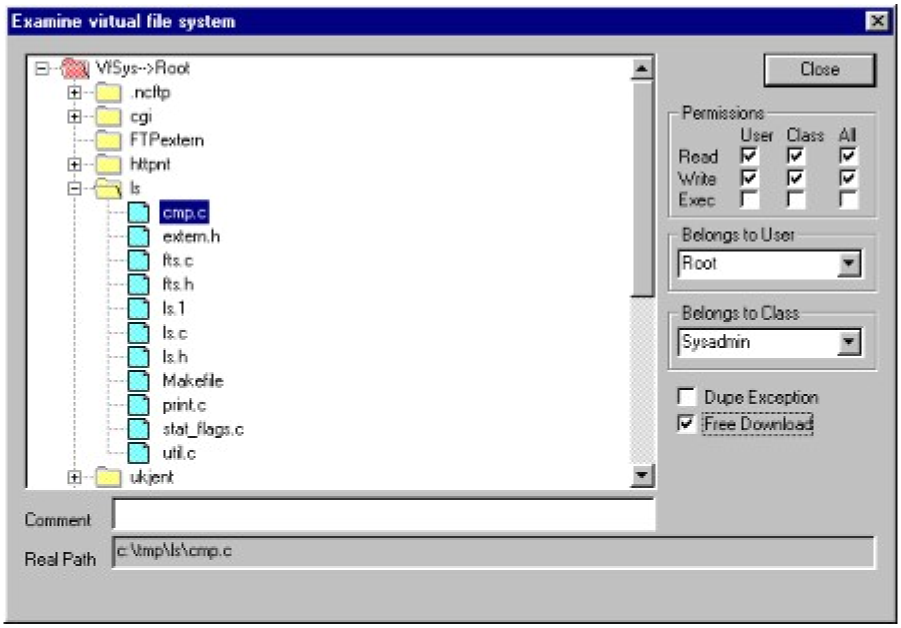
If there are many files in the Virtual File System path, the scanning of directories can take some time. The server will suspend all other operation and show a busy cursor until it is done. If you include CD ROM drives in the path you can expect this process to take several minutes.

See also: Virtual File System. and Virtual File System Options

View the Virtual File System

Toolbar:
Menu: Properties/Vfsys

From this dialog you can examine and alter the Virtual File System and it's files.



Permissions: Read ☑ User ☑ Class ☑ All; Write ☑ User ☑ Class ☑ All; Exec ☐ User ☐ Class ☐ All

Belongs to User: Root

Belongs to Class: Sysadmin

☐ Dupe Exception

☑ Free Download

Comment:

Real Path: c:\tmp\ls\cmp.c

Permissions

The permission are standard UNIX file permissions, divided into User, Class (in real UNIX it is called Group) and All (In real UNIX it is called Other).

When the Virtual File System checks the access permissions to a file or directory, it first check the "All" permissions. If that fails, it checks whether or not the user is a member of the owner class. If he is, it checks the permissions on the class. If the user still not has access, it checks if the user is the owner of the file, and again looks at the permissions.

Note: Users *in the Sysadmin class has the same rights as the UNIX "root"* user. *They override the permissions and gets full access.*

Read. Read permission to files and list permission to directories.

Write. Write/delete permission to files and write/create/mkdir/rmdir permissions if set on a directory.

Exec. Execute permission (a user can execute a .bat or .exe file via the SITE EXEC command) (- not yet implemented) if set on a file. On directories the Exec permission determines if the user has any access to the contents of the directory. If not set, the user will not be able to access any files or directory in the path in or below that directory.

Belongs to User

The owner of the file.

Belongs to Class

The class the file belong to (owner class).

Special attributes

Dupe Exception. Filenames with this flag set can be uploaded even if the dupe checker is active.

Free Download. Files with this flag set can be downloaded without affecting the users download restrictions. Such files will neither be added to the users download counter or to the users download statistics.

Comment

The comment field let you see and alter the comment of a file. The comment will show up in the .Index.txt and .SysIdx.txt files.

DOS Path

This filed shows the physical location of the file. It is for your information only.

See also: Virtual File System. and Virtual File System Options

Flush the Virtual File System

Menu: Properties/Vfsys

The Virtual File System maintains information about your files, like who that uploaded a file, how many times a file is downloaded, file comments and UNIX permissions. When changes are made it remember the change. Since this information not can be stored in the physical file system together with the filename, date etc., it will write it's own information in each directory that contain files with extra properties (files with comments, files that are uploaded or downloaded etc.). The filename it write this information to is named .Index.txt, but is *not* the same file that the user is presented for. In fact, the real .Index.txt that exist on the disk has the DOS/NT hidden flags et and will not show up in any directory listings.

If changes are made, the Virtual File System will flush the updated information to disk every 7. minute. It will also flush the information whenever it is stopped, and when the server shuts down.

The Flush *command* is provided to let you force an immediate flush. This can be useful if you work on utilities that will read or modify the physical .Index.txt files.

See also: Virtual File System. and Virtual File System Options

Start Service (go online)

Toolbar: ⚡
Menu: Properties

War FTP Daemon can operate in online and offline mode. When online, it will "listen" for incoming connections, and accept logins if no security parameters denies it. This is the normal mode of operation.

Stop Service (go offline)

Toolbar: ⚡
Menu: Properties

War FTP Daemon can operate in online and offline mode. When online, it will "listen" for incoming connections, and accept logins if no security parameters denies it. This is the normal mode of operation.

However, sometimes it can be useful to have the server running while it is offline. This mode of operation is first of all used for maintenance and configuration.

Note: The server will not shut down if you take it offline. The program will run, but it will disconnect from the network and not provide any services for incoming calls. Depending on the shutdown options it will either ask, or just break the connection and disconnect all current users.

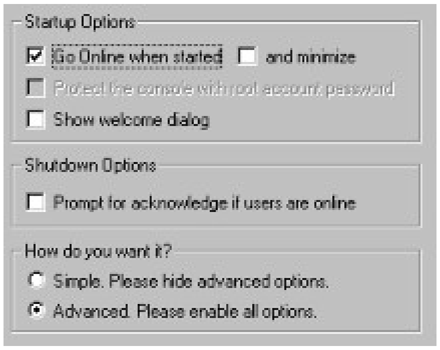
Options Tab

Toolbar: O
Menu: Properties/Options

The Options tab is where you configure the system wide settings for your FTP service

- General Options
- File System Options
- Virtual File System Options
- SITE commands
- FTP Options
- Server Name
- System Priority
- Sounds
- Upload Verification
- Log
- NT

General Options



Startup Options

☑ Go Online when started □ and minimize

□ Protect the console with root account password

□ Show welcome dialog

Shutdown Options

□ Prompt for acknowledge if users are online

How do you want it?

○ Simple. Please hide advanced options.

● Advanced. Please enable all options.

Startup Options

Go Online when started. If checked, the server will go online when it is started and wait for incoming connections.

... and minimize. If checked, the server will minimize itself. On Win95/NT4 systems it will disappear and only be available from the system tray.

Protect the console with root account password. If checked the system will ask for a "root account" password whenever you try to restore the window. In order to access the server console you will have to type in a user that belongs to the "Sysadmin" class and his password. If you don't have any users in this class, you will have to shut down the server from the system tray icon. Then edit the "FtpDaemon.ini" file and set the following value: `Protect Console=0`

Show welcome dialog. If checked, the welcome screen with copyright information will show up when the server starts up.

Shutdown Options

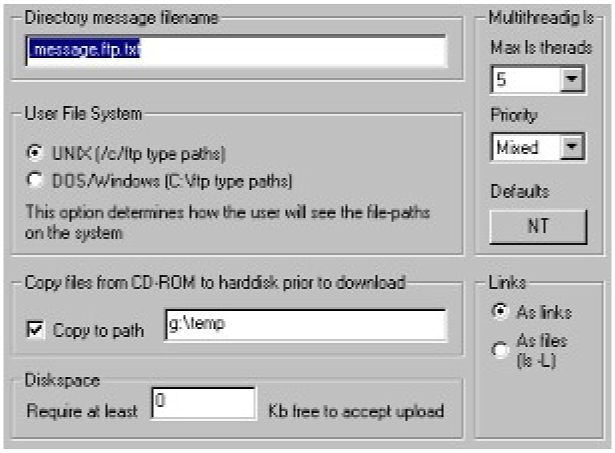
Prompt for... If checked, the system will ask for confirmation whenever you tell it to go offline or shutdown if there are users online. Note: If you use the SITE SHUTDOWN command, the server will shut down without checking this option.

How Do You Want it?

Simple. The server will operate as usual, but will hide most of the advanced (and maybe confusing) options. This mode is designed for users with small FTP sites that don't need all the advanced options. The mode only affect the security tab's.

Advanced. The server will enable all options and provide maximum power and flexibility.

File System Options



Directory message filename: .message.ftp.txt

User File System

● UNIX (/c/ftp type paths)

○ DOS/Windows (C:\ftp type paths)

This option determines how the user will see the file-paths on the system

Copy files from CD-ROM to harddisk prior to download

☑ Copy to path: g:\temp

Diskspace

Require at least 0 Kb free to accept upload

Multithreadig ls

Max ls therads: 5

Priority: Mixed

Defaults

Links

● As links

○ As files (ls -L)

Directory message filename

When the user issues a CD command and goes to a new directory, the server can answer with a little message about what he can expect to find in that directory. This is common practice on a large number of FTP sites, and most modern Windows FTP clients will understand the message and display it to the user in a separate window.

In order to enable this feature, create a file with the same name as the file in this box in each directory where you want these messages to appear. The server will see the file when the user changes to that directory and display the file. If you don't want these files to show up in the directory listings; set the DOS/NTFS hidden flag on the file.

There is also an option in the View/Messages dialog to set a default text to show if there is no directory message in the directory the user changes to.

User File System

This mode determines how the server will show file paths to the FTP user. Many DOS users will expect to see typical C:\whatever style paths, and feel familiar if the server use these. The problem is that many FTP clients and WEB browsers will be confused by this and misbehave. Therefore the server supports both DOS and more traditional UNIX paths.

Example on DOS paths

```
C:\
C:\FTP
C:\WINDOWS
```

Example on UNIX paths

```
/C
/C/ftp
/C/windows
```

The mode will only affect the output. The server will accept both DOS and UNIX paths from the user. He can even mix the paths, so something like: `/C/windows\temp` or `C:\windows/temp` are both valid paths.

If the Virtual File System is running the C:\ part falls out. The virtual file system don't use drives, as all the paths are put together in one hierarchy. The output will only differ in the use of slash (/) or backslash (\).

Note about DOS paths: The \ path will (if the user has \ as root-path) go to a logical directory behind the drives and list the drives on a directory listing

```
CD \
LIST

C:
D:
E:
```

Note about UNIX paths: Real UNIX paths on real UNIX systems are case sensitive. The file MyFile.txt and myfile.txt are regarded as two different names. The War FTP Daemon permits case insensitive filenames, and will treat `MyFile.txt` and `myfile.txt` as the same file.

Copy files from CD-ROM

CD-ROM's are getting more and more used on FTP sites. On a 'normal' FTP server, a CD-ROM drive will slow down file IO, the speed of directory listings. If you use a CD-ROM jukebox things will be even worse. Fortunately, the War FTP Daemon address this.

In order to have the same speed on directory listings on a CD-ROM or CD-Jukebox, use the Virtual File System. That way the contents of the CD-ROM is kept in RAM by the server, and a user can browse around without any physical access to the drive.

In order to gain the same speed on downloads from a CD-ROM as from a hard disks, enable the 'Copy to path' option, and specify a path that is inaccessible for the FTP users, and has free space to store large temporary files. When this option is enabled, all files that is to be downloaded from a CD-ROM (or network drive) is copied to the temporary path before the transfer start. If two users request the same file at the same time, it will be copied twice. When the transfer is done (or aborted), the temporary file is deleted.

The server use a dedicated thread for the copying. This means that all operations on the FTP site will continue at full speed, while a file is being copied. It also guarantees that only one file is copied from a CD-ROM drive (or jukebox) at the time - giving the best overall performance.

Note: If a copy operation fails, the file will be transmitted directly from the CD-ROM. This fallback feature ensures that users will get their files, even if you run out of temporary storage space on the harddisk.

How To Display Links

War FTP Daemon emulates standard UNIX FTP servers. The directory listings are in the same format as the UNIX ls command would have given.

In UNIX links are displayed like `LinkName -> RealName`, with the 'l' flag set. This can confuse many FTP clients, since there is now way to know if the name represents a file or a directory. The UNIX ls command has an option (-L) that will display the link as if it were a normal file or directory. The server supports this option, but many FTP clients don't use it.

To overcome these problems, you can set the -L option as default, forcing the deamon to output any link as if it were a normal file. This option is by default turned on.

Diskspace

This option will deny upload if the available free diskspace on the target partition/drive is less than the KB’s defined.

Multitreading ls

“Ls” refers to the UNIX command ls (list structure). This is because War use the ls output format and command syntax for the LIST command.

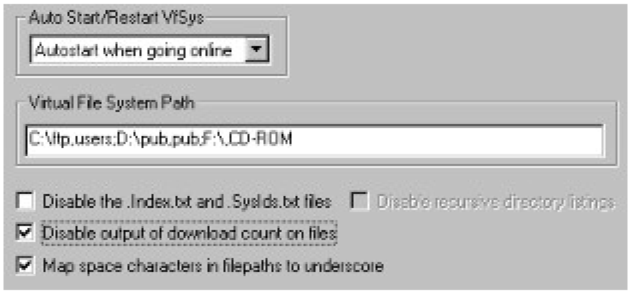
Unlike most other Windows FTP servers, War will not stop the processing of ongoing file transfers to wait for slow IO calls in order to generate a directory listing. In stead it will pass the LIST command to another thread and continue to process basic FTP operations. You can specify the maximum number of threads that will be created for this function. Currently there is a limit of 5 threads. This is sufficient to server several hundred FTP users under normal conditions. A LIST command will typically complete in less than 1/5 second - but sometimes it can take longer. If users LIST CD-ROM drives, or perform recursive listings of a large number of files/directories, the processing can take several minutes (works case).

War has logic to pass new LIST commands to idle threads, or to the tread that will most likely finish the current task first.

If the listing is huge, the processing of a directory listing can not only be slow, but also take quite some CPU time. Under NT the overall performance of the server will be better if the priority of the ls threads are decreased when this happen. The Mixed Priority option will decrease the priority when War is processing data or waiting for slow IO, and increase when it access shared memory in the Virtual File System or the User database. Windows 95 has a considerable overhead when a process change priority, so this mixed priority will typically make it perform 100 - 1000 times slower than the static option. If you select “Optimal” War will choose the optimal setting based on the current environment.

If you don’t know what “threads” and “priority” are, just press the NT or Win95 button and let War decide the best settings for your machine.

Virtual File System Options



Auto Start/Restart VfSys

Autostart when going online ▾

Virtual File System Path

C:\ftp,users;D:\pub,pub;F:\,CD-ROM

☐ Disable the .Index.txt and .SysIds.txt files ☐ Disable recursive directory listings

☑ Disable output of download count on files

☑ Map space characters in filepaths to underscore

Auto Start/Restart option. If you use the Virtual File System you will usually want the server to start it automatically. You can have War start it, and also re-scan the physical file system at regular intervals to update the information in case files are added/deleted by other applications. The rescanning will only take place when no FTP users have open handles to the virtual file system.

Virtual File System Path. This is the path(s) that the virtual file system will cover. When running the Virtual File System, only files inside the path's defined here will be available.

The formal path definition is as follows: X:\path[path][,pseudo-name][;new path definition]

Let's have some examples. You have a D: drive that you want to use for your FTP service. All files on this drive shall be available to all (or some of) the users of your site.

```
D:\,ftp
```

The path above will include all files on your D: drive in the Virtual File System, and give the user a root catalog with one directory, /ftp. Your D: drive is now mapped to /ftp.

The reason that paths should be mapped is that the Virtual File System not can resolve the root (/) path to any DOS paths. It can resolve all the files or directories in the root directory, but can not create any new files here (except links). The servers root path is therefore always read-only. If you create a pseudo-name for your drive (or path), that entry can be resolved to a DOS destination, and write permission is available.

Another example. You have some directories spread over your physical drives that you want to share with your users.

```
C:\FTP\USERS,usr;C:\TEMP,tmp;D:\MISC\INCOMING,incoming;E:\MISC,pub
```

The path above will create the following directory listing in the root directory:

```
/incoming
/pub
/tmp
/usr
```

Important: You can NOT define the same physical path more than one time. `C:\,c_drive;C:\WINDOWS,win` is an example on an illegal Virtual File System path. The Windows directory are defined twice. The server will currently not detect such errors, but might crash or misbehave. *It is your responsibility to make sure that no directories are defined more than one time*. If you want a directory to show up several times, use links.

Note that all files and directories below the DOS paths given in the Virtual File System Path will be *included.*

Disable the .Index.txt and .SysIdx.txt files. This option will make these files unavailable.

Disable recursive directory listings.

The directory listings generated by the War FTP Daemon are in UNIX *ls* format. In fact, the server will emulate most of the options to the UNIX ls command also. One of these options are -R. If a user issues a LIST -R or LS -R command to his FTP client, War FTP Daemon will generate a recursive listing of all directories and files, starting in the current directory.

Some FTP mirroring software depend on this feature in order to get a full lusting of all the files on a site. However, scanning thousands of files takes time and memory, and if a lot of users are issuing this command the performance of your server will drop. The output from a large FTP site can be several megabytes.

If you don't want to spend CPU time and memory on this, don't need the .Index.txt and .SysIdx.txt files and don't need to support FTP mirroring software, - you can turn this option off.

This option can not be turned off if unless the .Index.txt and .SysIdx.txt files are disabled.

Disable output of download count...

When the virtual file system is running it will maintain a download counter on each file it knows about (that resides in directories with DOS write access - CD-ROM's can not use this benefit). It will output this counter as the first statement in the comment of each file in the .SysIdx.txt file if this option is enabled.

This option is only relevant if the .Index.txt and .SysIdx.txt files are enabled.

Map space...

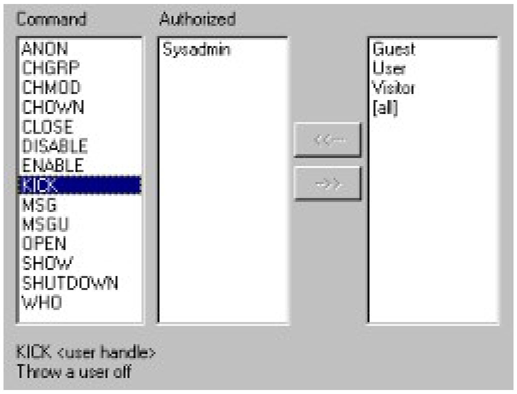
When the FTP protocol was designed, space was generally not used in filenames. The protocol can handle it, but many FTP clients and WEB browsers will be confused and misbehave. In order to avoid this, you can map all spaces in filenames to underscore.

Example:

`Program files` will map to `Program_files.`

See also: Virtual File System.

SITE commands



KICK <user handle>
Throw a user off

The FTP protocol defines all valid commands a FTP user can send to the server. Some servers however has extra services to offer. This is done via the SITE command. SITE commands are system specific, and not compatible from server to server.

War FTP Daemon offers some useful commands to the user this way, and also a high level of security. Only users in classes defined in the SITE commands tab are allowed to execute the commands. The permissions are granted on a per-command level.

Security

In order to allow or disallow the users in a user-class access to a command, simply click on the command and then move the class name from or to the Authorized box. If you move the [all] tag to the Authorized box, all users will get access to the command.

Supported commands

MSG <any message>
Sends a message to the console

CHOWN <user name> <file or pattern
Changes the owner of a file/files

CHMOD <octal mode> <file or pattern>
Changes the permissions of a file/files

CHGRP <class> <pattern or file>
Changes the group (class) of a file/files

WHO
Lists the users online

MSGU <user handle> <any message>
Sends a message to a user

SHUTDOWN
Shuts the system down

DISABLE <USER | GROUP | CLASS> <account name>
Disables an account

ENABLE <USER | GROUP | CLASS> <account name>
Enables an account

KICK <user handle>
Throw a user off

CLOSE
Close the system for everyone but system operators

OPEN
Open the system for general access

ANON <ALLOW | DENY>
Allow or deny anonymous access

Examples

From the FTP client supplied with Win95 and NT

```
QUOTE SITE MSG hi there, how is things?
```

This command will send the message "how is things?" to the Messages from the Users window on the system console.
The "QUOTE" command tells the FTP client to pass the rest of the line to the server. The "SITE" command tells the server to interpret the next word as a command.

QUOTE SITE SHUTDOWN

This command shuts the system down.

QUOTE SITE WHO

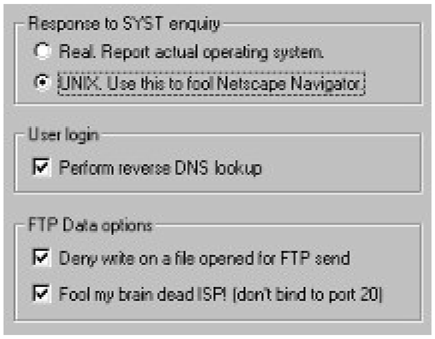
This command returns a list of the users online. The leftmost filed in the listing is a handle number for the connection. You can use this handle number to kick users off the system.

QUOTE SITE KICK 23

This command kick user #23 off the system. He will be disconnected at once, and any file transfer in will be aborted. Note that the number is the user *handle*, as return by the WHO command.

For other FTP clients, please refer to their documentation or help file.

FTP Options



Response to SYST enquiry

○ Real. Report actual operating system.

● UNIX. Use this to fool Netscape Navigator.

User login

☑ Perform reverse DNS lookup

FTP Data options

☑ Deny write on a file opened for FTP send

☑ Fool my brain dead ISP! (don't bind to port 20)

Response to the SYST inquiry

SYST is one of the commands defined in the FTP specifications. The response shall be an operating system identification code as defined in RFC943 (or the replacement of this document), section "Assigned Numbers". FTP clients and WEB browsers use this command to determine the format of the paths and directory listings they can expect.

War FTP Daemon is a Win95/NT application, and the formal correct response is WIN32. However, only a few FTP clients will accept this response, and then recognize the UNIX style directory listings. Netscape Navigator 2.1 goes gaga if the server answer it's real operating system. Therefore the server has the option (enabled by default) to identify itself as a UNIX system.

User login

War FTP Daemon can perform a “reverse DNS lookup” when users log on. This will bring the users IP name up when you spy on the user.

Some UNIX FTP servers will deny login from users if the reverse DNS lookup fails. Currently War does not support this restriction, as it does not give any kind of protection or security. Also, sometimes the lookup will fail on computers that have a valid DNS name.

The implementation of the Reverse DNS lookup in war is asynchronous. The user can log in and use the server while War is waiting for the DNS servers to resolve the name.

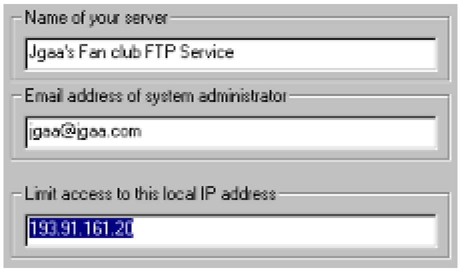
If War is unstable, you might turn this option off as it has been suggested that it might be the cause of some problems on some computers. The reverse DNS lookup has no meaning for the server itself. It’s just a “fancy” feature that has been requested by a huge number of War FTP Daemon users.

FTP Data options

Deny write on a file opened for FTP send will place a lock on the file, preventing other applications from writing to the file until the transfer is completed.

Fool my brain dead ISP will prevent the server from binding the data socket to the local port 20. This will make some problems if the server is located behind a firewall that allow FTP access, but it will fool the filters brain dead ISP's use to stop customers from setting up FTP servers. If you are unfortunate enough to have such an ISP the best is to cancel your account and get a new account with a more friendly company. If you are stuck with your ISP, you can enable this option and specify another port on the servers main console (port 2121 is usually a good choice).

Server Name



Name of your server: Jgaa's Fan club FTP Service

Email address of system administrator: jgaa@jgaa.com

Limit access to this local IP address: 193.91.161.20

War FTP Daemon supports macros in all messages sent to the user, including welcome messages, directory change messages etc. In order to make it easy to change manifests such as site name and email address of the system operator, you can define this here.

Name of your server. Maps to macro: [$systemname]

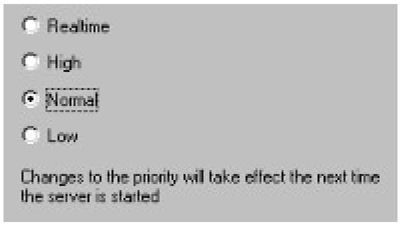
Email... Maps to macro [$email]

Limit access...

This option is usually used with multimomed (virtual) servers. If this field is in use, the server will only "listen" to the specified IP number. If this filed is blank, the server will listen to all IP numbers known by Winsock.

Note: If you don't know what multihoming is, leave this field blank.

System priority



○ Realtime

○ High

● Normal

○ Low

Changes to the priority will take effect the next time the server is started

Realtime. The highest possible priority. The threads of a real-time priority class process preempt the threads of all other processes, including operating system processes performing important tasks. For example, a real-time process that executes for more than a very brief interval can cause disk caches not to flush or cause the mouse to be unresponsive.

High. Indicates a process that performs time-critical tasks that must be executed immediately for it to run correctly. The threads of a high-priority class process preempt the threads of normal or idle priority class processes. An example is Windows Task List, which must respond quickly when called by the user, regardless of the load on the operating system. Use extreme care when using the high-priority class, because a high-priority class CPU-bound application can use nearly all available cycles.

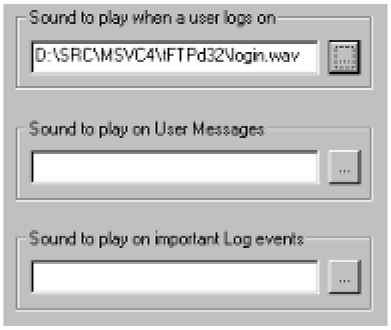
Normal. Indicates a normal process with no special scheduling needs.

Low. Indicates a process whose threads run only when the system is idle and are preempted by the threads of any process running in a higher priority class. An example is a screen saver.

The default priority is *Normal*.

Note: Changes to the priority will not take effect before the server is restated. Just going offline and online will not affect the priority. The server sets the priority when it starts up. The reason for this is to ensure that all the threads gets the same priority.

Sounds



Sound to play when a user logs on

D:\SRC\MSVC4\IFTPd32\login.wav

Sound to play on User Messages

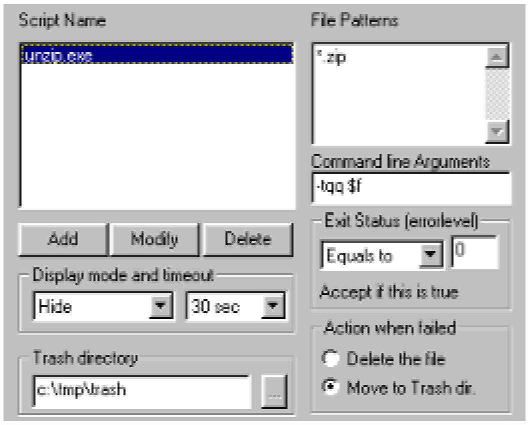
Sound to play on important Log events

War FTP Daemon can play sounds when some events occur. It can be especially useful to play a sound to notify that user messages has arrived.

Note: On some systems the sound feature can cause the system to crash when the server tries to play a sound. In general it seems to work fine on Win95 and NT4.0 (provided that the server runs as a user application and not as a system service).

The supplied .wav files is copyright Joe DeShon and is *not* in public domain! They might be used and distributed with this software. Any other distribution is denied unless otherwise agreed by Joe DeShon. Check out http://www.sky.net/~jdeshon/joewav.html for details.

Upload Verification



Script Name: unzip.exe

File Patterns: *.zip

Command line Arguments: -tqq $f

Exit Status (errorlevel): Equals to 0

Accept if this is true

Display mode and timeout: Hide, 30 sec

Trash directory: c:\tmp\trash

Action when failed:

○ Delete the file

● Move to Trash dir.

One interesting feature of UNIX FTP servers, that are rarely found on the Windows platform, is the capability of verifying or processing incoming files after the upload is complete.

War FTP Daemon has this option. When a file is uploaded it checks to see if the filename matches one of the patterns to any of the Script names defined in the Upload Verification tab. If a match is found, the script is started, and based on the Exit value of that script, the server determines if the upload will be accepted or not.

This feature can be used to perform CRC checks, virus scan, or general processing, like extracting file_id.diz from an archive.

When a script is started the user that uploaded the file will have to wait for the result from the script processing. Any other users will continue theirs tasks as normal.

Script Name

The script name is the name of an executable DOS or Windows program. *If you start a Windows program to process a script, make sure that it supports running and processing the line arguments without any user input.*

When the script (program) is started, it's current directory is set to the directory where the incoming file was uploaded.

File Patterns

You can specify one or several patterns to look for. The pattern applies for the actual DOS name of the file, including the full path.

If you want to do something with all incoming .zip files, simply enter `*.zip`
If you will be processing several file types by one script, you can enter more patterns, each on their own line.

Command line Arguments

The script will be started with the command line arguments specified here. You will normally pass over the name of the uploaded file and some line switches to tell the script how to process the file.

If you use UnZip.exe to CRC check incoming zip files, the argument will be:

```
-tqq $f
```

-tqq is command line options for UnZip.exe and instruct it to test (not extract) the archive in quiet mode. $f is a special macro that represent the uploaded file name

The following macros are available:

$f	Filename
$u	User name
$p	File name, including the full DOS path
$d	Directory name where the file is uploaded
$g	Group name of the user
$c	Class name of the user
$$	Just a single $

Exit Status

When a program finishes processing and terminates, it returns a status value to the operating system. In DOS .bat files this value is represented by "errorlevel". When the server starts a script, it monitors its processing, and receives the exit status when the program terminates. This value is used by the server to verify the upload.

Any well written file verification program will return well documented values. The usual approach is to return 0 if everything went OK, and a negative or positive value to identify problems or errors. The server does not mind *what* that went wring, all it is concerned about is if the file was OK.

You can specify a return value to check for, and assign it to one of the following rules to accept the upload.

Equals to
Not
Less than
Greather than
Ignore

If you specify *Ignore*, the server will start the script and accept the upload regardless of the Exit Status. This mode is added to allow processing of incoming files without any verification. I.e. If you are waiting for some file to be uploaded, you can call wSendmail and notify yourself or others via email about the file.

Action when failed

If the verification fails, the server can either delete the file, or move it to a special directory for manual verification at a later time. If you choose Move and the file already exist in the thrash directory, the file will be renamed in stead of moved.

Trash directory

A DOS path to a directory to place files that failed verification. This property is shared among all the scripts.

Display mode and timeout

The display mode determines how the server will start the script.

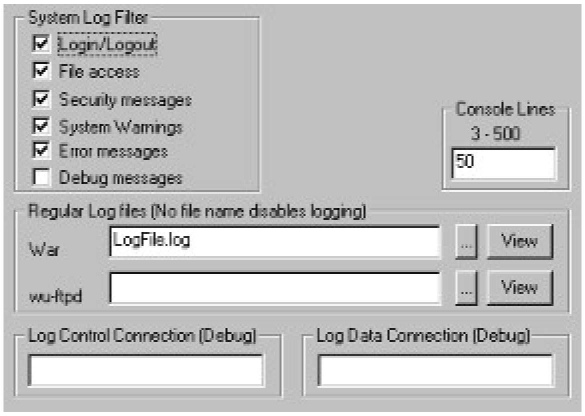
Hide (The default mode)
Minimize
Normal
Normal/Focus

The timeout tells the server how long it will allow the script to run before it is terminated by the server. Since these scripts are used to process incoming files, and the FTP user will be held until the script terminates, they can not run for too long. If a script runs more than 20 seconds, it is typically because it want input from the user. In such cases the server simply regards the script as unreliable, terminate it when the timeout has expired and accept the upload.

Note: The timeout value only tells the maximum time a script is allowed run. When a script terminate normally, the server immediately suspends the held state of the user and either accept or deny the upload.

Note: There are written several scripts and programs by War FTP Users that use this feature. Some of these utilities are released as Freeware and is available from http://www.jgaa.com

Log Options



System Log Filter

☑ Login/Logout
☑ File access
☑ Security messages
☑ System Warnings
☑ Error messages
☐ Debug messages

Console Lines 3 - 500: 50

Regular Log files (No file name disables logging)

War: LogFile.log

wu-ftpd:

Log Control Connection (Debug):

Log Data Connection (Debug):

This dialog allow you to filter the log messages you want to see. If you specify a log filename, the log will be written to disk. If not, the log will only appear in the log window on the system console.

System Log Filter

Login/Logout. All events regarding logins and logouts.

File Access. Files uploaded, downloaded, deleted, renamed. Also directories that are created or removed.

Security messages. Users that are denied access, bad passwords etc.

System Warnings. Non-critical error situations in the different modules of the server.

Error messages. Critical error situations.

Debug messages. Generates a huge number of log output. Used to trace problems in the server or FTP clients.

Log Files

War: A filename where the server can store the log. This file is flushed and closed each second, so that other programs can view the file.

Note: If the server fails to open the file, the log events in the output queue will be trashed and only appear on the console.

Wu-ftpd: This is an optional log file that will log all successful file transfers in the same format as the popular wu-ftpd UNIX daemon. This log format is easier to parse in order to generate reports from the log files.

Log file name macros

The log file names can contain macros. This is convenient if you want to start on a new log file each day. The valid log file macros are:

%a	Abbreviated weekday name
%A	Full weekday name

%b	Abbreviated month name
%B	Full month name
%d	Day of month as decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01 – 12)
%j	Day of year as decimal number (001 – 366)
%m	Month as decimal number (01 – 12)
%M	Minute as decimal number (00 – 59)
%p	Current locale's A.M./P.M. indicator for 12-hour clock
%S	Second as decimal number (00 – 59)
%U	Week of year as decimal number, with Sunday as first day of week (00 – 51)
%w	Weekday as decimal number (0 – 6; Sunday is 0)
%W	Week of year as decimal number, with Monday as first day of week (00 – 51)
%y	Year without century, as decimal number (00 – 99)
%Y	Year with century, as decimal number
%z, **%Z**	Time-zone name or abbreviation; no characters if time zone is unknown
%%	Percent sign

Log Lines

The number of lines that will be stored in the Log Window on the system console. When the log exceeds this number the oldest line will be removed.

The default number is 50.

This option only appear for the screen. The log files will be written to any length.

Note: If the number of lines is larger than the window can handle, the output on the console can look corrupt. If this happens, decrease the number of log lines.

Log all data xmitted over the control channel.

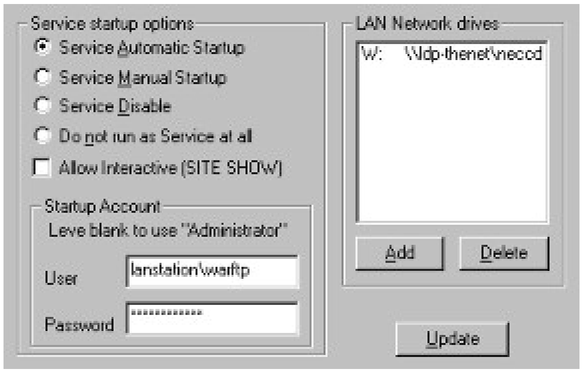
If a filename is specified here the server will dump all data sent or received over the FTP control connection to that file. This is usually only done when tracing problems in the server or FTP clients.

Log all data xmitted over the data channel.

If a filename is specified here the server will dump all data sent over all active FTP data connections (a copy of all file transfers). This option should be used with caution, as the file will grow huge. Also, if several users are online when this option is enabled, the data dumped will contain a little from one transmission and a little from another transmission, and make very little sense.

NT Options

This chapter does only apply to the server when running under Windows NT



Service startup options

● Service Automatic Startup
○ Service Manual Startup
○ Service Disable
○ Do not run as Service at all
□ Allow Interactive (SITE SHOW)

Startup Account
Leve blank to use "Administrator"
User lanstation\warftp
Password *************

LAN Network drives
W: \\dp-thenet\necod

Overview of a NT service

When Microsoft designed Windows NT, they decided to implement a special runtime-mode for programs that provided services to the machine/network, rather than services to a single user. They invented the NT Service API that allows a program to be started when NT starts up, and lives more or less as part of NT itself, independent of users that might log on and off the machine. All disk drivers, printer drivers, Network drivers, RAS etc. etc. runs as NT services.

A NT service is not supposed to interact directly with the user. Normally you will configure a service from an applet in the control panel.

War FTP Daemon as a NT service

War FTP daemon has native support to run as a NT service. When it is configures to run as a service, it will normally start when NT boots, and remain running until NT is shut down. All resources used by War is released when it stops (I mention this since some NT services does not free memory and dll’s they have used).

Unlike most NT services War was designed as a true Win32 program, using MFC, message maps and asynchronous operations. This makes war run very fast, using few threads and limited resources. However, it needs some windows for message passing between the submodules. The NT Service implementation was never design for this kind of programs. A typical NT service is a 32 bit console application with no windows and no message pump, using only low-level API calls. The modern design of War have therefore caused a few problems when running as a service. See the release notes for a list of known problems, and how to avoid/handle them.

Service startup options

This is basically the same options that you will find in the “startup..” dialog in the Service applet in the control panel.

Note: If War is configured to run as a service, it will take about 30 seconds to start up when you start the program manually as a normal program. This is because it first checks with NT if it is supposed to run as a service or as a normal program.

If you want to use another account than the local “Administrator”, the following rules apply:

- The user account must have “login as a service” permission.
- The user account must have “Administrator” rights. I don’t know why, but the service will freeze if it don’t have these rights. I have not been able to communicate with the program at all when testing this. War fails to output status information to files or even to start the debugger from the DebugBreak() call.
- The user ID must be given as machinename\userid, ie. “lanstation\warftp”
- The password must be given if you specify a user.
- If you use an account for the service, this should be a dedicated user account, and not the account of one of the regular users on the network/machine.
- You **must** create an NT user account for War and start it with this account if you want to use the automatically mounting of network drives option.

Note: You must press the button in order to activate the new Service options. If an error occurs you will be noticed.

Interactive option

There are two ways to configure War when it runs as a NT service. The recommended method is to

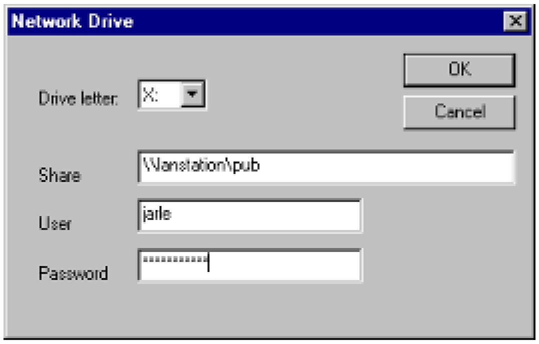
1. Stop the service from the Control Panel
2. Start the program as a normal application
3. Do the configurations
4. Exit the program
5. Restart as a service from the Control Panel

The other method is to enable the “Interactive mode”. This will allow you to issue a SITE SHOW command from a FTP client. When this command is received War will open it’s usual GUI as a normal window, and you can access everything but the NT tab in Options. When you are done you minimize the window to make it disappear.

Note: This option is convenient, but it has one huge drawback. There is a bug in MFC that will make War crash when you log off or reboot NT. It seems like NT will close/destroy an internal hidden window created by MFC to communicate with the CAsyncSocket() class. There is no known work-around for this problem. It only happen when the Interactive option is enabled. You can use the Interactive option if you want War to start when NT boots, but you must manually shut War down before you log off NT or reboot the machine.

LAN network drives

Since War will start when no users is logged on in service mode, it has the capability to mount one or several network drives. This is useful if the FTP users are supposed to access files on the local network (LAN). War use generic functions that should be able to access both Novell and NT shares. It will not be able to access shares that have password security. When the shares are mounted it will try to log on to the drive as the specified user with the given password.



Drive letter: X:

Share \\lanstation\pub

User jarle

Password ***********

You should ***not*** use a drive letter that any users will try to assign to network shares when they log on. This is because the Drive Letter and the share will be mounted as if it was a physical disk drive on the machine. All users logging on to the NT machine will be able to access the drive as if it was a local drive, with the permissions of the user specified in the dialog above. **This is a security hole you should be aware of.**

Note: This feature will only be reliable if you start War as a dedicated NT user as stated above.

The Virtual File System

Introduction

The War FTP Daemon can optionally read the directory trees from one or several drives or paths into memory, and use it's internal copy of the directory structure in stead of accessing the physical disk whenever it needs to check a pathname or make a directory or file listing. On a system with many users online, this feature will give a significant performance improvement. The virtual file system should be used on any static download service. (Files uploaded, renamed and deleted by online users will be handled, files copied from the operating system will not be found until the virtual file system is restarted).

With the virtual file system you also have the opportunity to design your own custom file system layout, that can contain directories and drives (including network drives), to make it easy for the users to find the files they are looking for. Drives and directories not included in the virtual file system will be 100% secure, as the server wont even try to look them up.

The virtual file system is a partial implementation of a UNIX file system that is mounted between the physical file system (FAT, HPFS, NTFS) and the FTP file IO functions. You don't need to run the virtual file system. For most sites it will only take up computer memory and add more complexity to the site administration. But for advanced FTP site administrators it will give more power and flexibility than any other FTP server running under Windows.

FTP security and path mapping
|
Virtual File System: UNIX security and links, dupe checker and free download checker.
|
DOS/HPFS/NTFS

*Please note that when you use the Virtual File System; both UNIX **and** the FTP security functions are used. In other words; - the* user *must have **both** UNIX **and** FTP permissions in order to access a file or directory. See the graphical layout of the File Security*

The View Virtual File System dialog will allow you to modify the file/directory permissions. If you don't have a working knowledge on UNIX - don't play with the `rwx` flags! Use the right mouse button to create links, or simply add shortcuts with the Win95/NT4 explorer. Shortcuts to files that exist within the Virtual File System are automatically converted to UNIX links when the Virtual File System starts up, and the .lnk extension removed. (The original shortcut files are left untouched on the disk but are invisible to the FTP user).

UNIX commands

Stripped versions of the following UNIX commands are available trough the SITE command:

chmod   CHMOD mode FilePattern
The mode is the octal UNIX file permissions.

chown   CHOWN NewOwner FilePattern

chgrp   CHGRP NewGroup FilePattern
The UNIX group name is equivalent to the FTP class name. Users in the class Sysadmin has UNIX root user privileges.

File Pattern can be a file name or a pattern.

See also Virtual File System - Restart, Stop, View, Flush and Options.

Chmod (SITE command)

chmod mode file or pattern

Description

The chmod utility modifies the file mode bits of the targeted files as specified by the mode operand.

Modes

Modes must be absolute. An absolute mode is an octal number constructed by or-ing the following values:

0400	read by owner
0200	write by owner
0100	execute (or search for directories) by owner
0070	read, write, execute/search by group
0007	read, write, execute/search by others

The read, write, and execute/search values for group and others are encoded as described for owner.

Example

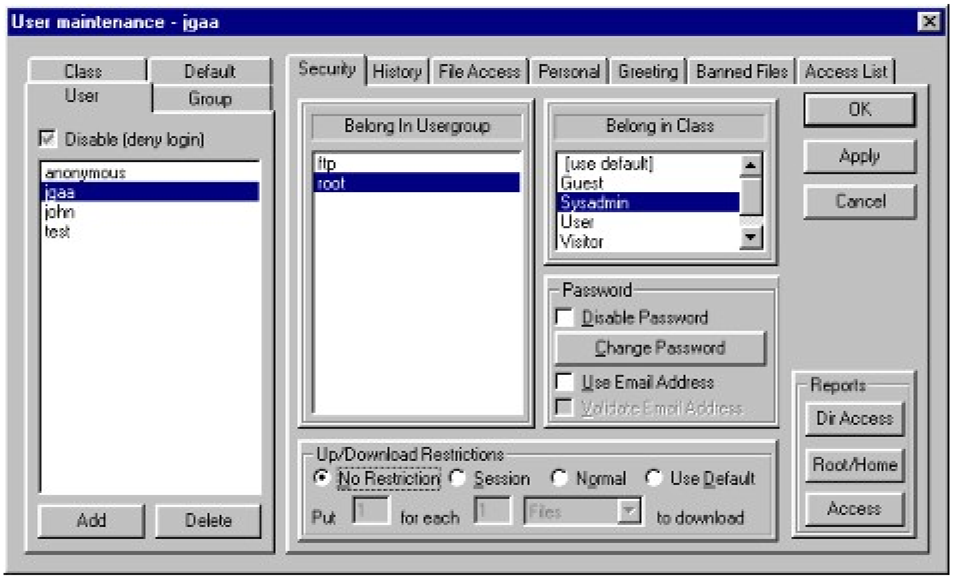
```
QUOTE SITE CHMOD 744 *.EXE
QUOTE SITE CHMOD 700 MYFILE.TXT
QUOTE SITE CHOWN JGAA *.TXT
```

Security tab

Toolbar:

Menu: Properties/Security/*

The security tab is used to edit user, group, class and the default system configuration. Some commands, such as the Properties/Security/Edit User, Group, Class and default menu items, will cause the server to hide the other tab's on the left side of the screen. Also, if you use the simplified mode of operation, most of the right side tabs and many of the options will disappear.



User maintenance - jgaa

☑ Disable (deny login)

anonymous
jgaa
john
test

Belong In Usergroup: ftp, root (root selected)

Belong in Class: [use default], Guest, Sysadmin, User, Visitor (Sysadmin selected)

Password
☐ Disable Password
☐ Use Email Address
☐ Validate Email Address

Up/Download Restrictions
● No Restriction ○ Session ○ Normal ○ Use Default

Put 1 for each 1 Files to download

The User tab

Adding and deleting users

Password

Up and Download restrictions

Up and Download counters

Idle time and max simultaneous logins

Login counter and the fail limit

Directory access permissions

Real name and memo

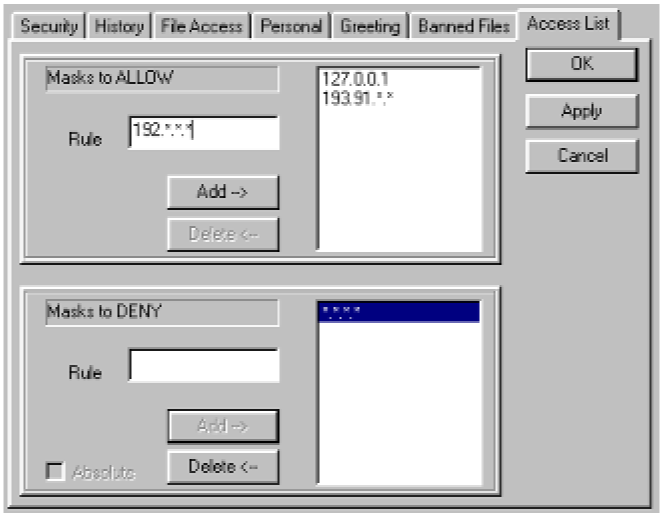
Greeting message

Banned files and the dupe checker

IP access restrictions

IP Access Restrictions

Dialog: Security tab Banned Files



Masks to ALLOW

Rule 192.*.*.*

127.0.0.1
193.91.*.*

Masks to DENY

Rule

..*.*

□ Absolute

Most people are nice. Some are not. If you know the IP number of some people you don't want on your system, you can mask them out.

Masks to DENY

Any IP address or IP mask defined here will be denied access, unless it is listed in the ALLOW list. If you want to restrict access for system administrators to the machines they use, you simply deny access for all addresses in this field.

Masks to ALLOW

Here you list exceptions from the deny list. If a IP address is denied, the system looks in the ALLOW list for an exception.

When the system checks the IP access lists, it first scans all 4 levels for a mask matching the calling users IP address. If it turns out that the user IP is denied (at any level), it scans the ALLOW lists at all 4 levels. If an exception is found (at any level) the user is allowed access.

How it works

The scanning takes place after the user has given his password (or after the password check is skipped if the user don't need a password), unless the □ Absolute option is checked on the default level. In this case the server will deny any IP number banned at the default level by simply closing the connection. If the user makes it trough this first test, the user, group and class levels will be checked after he has given his password.

Adding a new rule

To add an new rule, enter the IP address (or mask) in the Rule field. When the Rule qualifies as a valid IP address or mask, the Add button will be enabled. Press the Add button to add the new rule.

An IP number consist of four numbers 0 - 255, separated with '.'

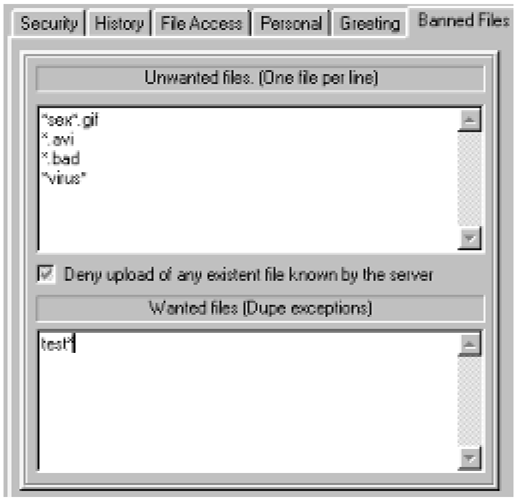
127.0.0.1
127.34.56.230

A Mask can be any valid IP number. Each of the for numbers can be a number, a range or a star, specifying range 0 - 255.

..*.*	All IP numbers
127.*.*.*	All ip numbers starting with 127.
127.0.0.1	The IP number 127.0.0.1
127.0.0.1-15	IP numbers in the range 127.0.0.1 - 127.0.0.15

Banned files and the dupe checker

Dialog: Security tab Banned Files



Unwanted files. (One file per line)

```
*sex*.gif
*.avi
*.bad
*virus*
```

☑ Deny upload of any existent file known by the server

Wanted files (Dupe exceptions)

```
test*
```

It is nice when kind people upload useful files to the server. But there might be some files you don't want, or some many you have your files spread in a large number of directories and don't want the users to waste time and bandwidth uploading duplicates.

The War FTP daemon has two ways to help you out.

Banned Files

If there are certain files you don't want, and you know the file names or can use some file pattern to identify them, add the files to the banned files list. This list can be defined on all 4 levels. When a user want to upload a file the server merges together a list of the banned files at all 4 levels (user, group, class, default) and check the filename against this combined list.

Dupe Checker

The dupe checker is one of the real powerful features of this server. If you run the Virtual File System you can enable the ☑ Deny upload of any existent file known by the server option. The server will then scan all the filenames in the Virtual File System, and if a name exist, deny the upload.

If the Dupe Exception flag is set on the file, the file can be uploaded even if the dupe checker is active.

This option can be ☑ yes,
□ no and
☑ default. If set to default, the server will look for a yes or no on a higher level. This permits you to enable/disable the dupe checker on groups, classes or the entire user database, provided that the option is set to default on the levels below.

Wanted Files (Dupe exceptions)

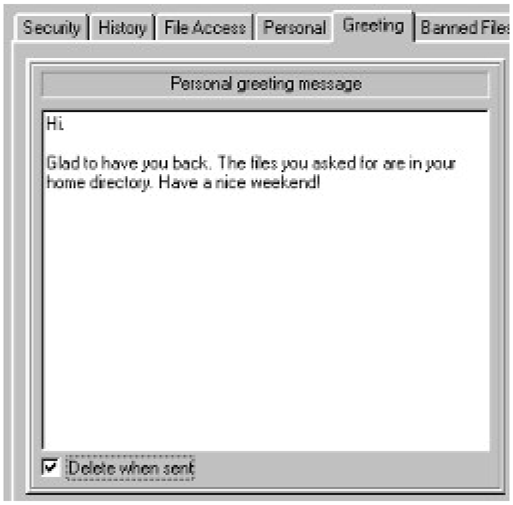
You can use the Dupe Exception flag to allow certain files to override the dupe checker, or you can specify filenames or file patterns to be allowed in this list, even if they exist.

This list can be defined on all 4 levels. When a user want to upload a file the server merges together a list of the wanted files at all 4 levels (user, group, class, default) and check the filename against this combined list.

Note: Banned files can not be overridden. If a file is identified as a banned file, the wanted list is not checked. The wanted file list is only checked when the dupe checker is active, and only applies for files that exist on the system and else would have been denied.

Greeting message

Dialog: Security tab Greeting



Hi

Glad to have you back. The files you asked for are in your home directory. Have a nice weekend!

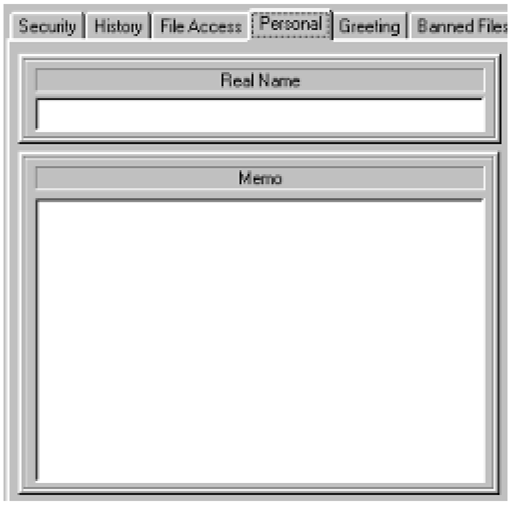
☑ Delete when sent

When the user has logged in, after the system welcome message has been displayed, you have the option to display a personal welcome message to the user. This can be a static message, or you can set the delete when sent flag to only display it once.

If messages are created on several levels (user, group, class, default), *all* the messages will be sent.

Real name and memo

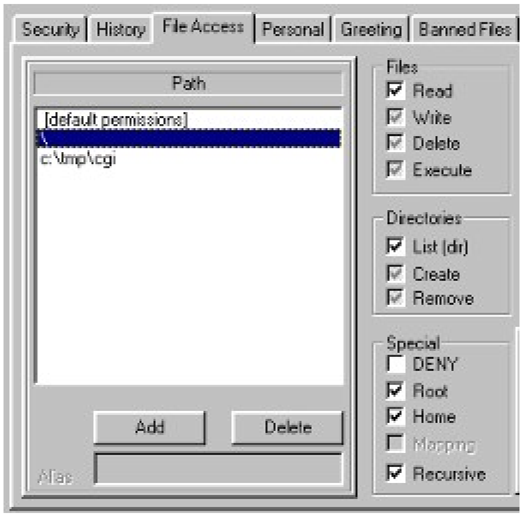
Dialog: Security tab Personal



These properties are for your convenience only.

Directory access permissions

Dialog: Security tab File Access



[default permissions]
\
c:\tmp\cgi

Files
☑ Read
☑ Write
☑ Delete
☑ Execute

Directories
☑ List (dir)
☑ Create
☑ Remove

Special
□ DENY
☑ Root
☑ Home
□ Mapping
☑ Recursive

The War FTP Daemon has a sophisticated Directory Access scheme. These properties can give a user access to or deny him access to any directory on your system, or within the Virtual File System.

If you use the Virtual File System, these properties comes on top of the UNIX file permissions defined in the Virtual File System. See the File Access Security table for a better understanding of the interaction between these properties and the Virtual file System.

States of the attributes

Many of the file attributes can have three states;☑ yes,
□ no and
☑ default. If a file i.e. is requested for download, the server will perform the following test (when the up/download checker, dupe checker and Virtual File System has all permitted the operation):

1. See it the user has read access to the files in that directory.
2. If the read attribute is set to default, check the users[default permissions] read attribute.
3. If that too is set to default, check the group's [default permissions] read attribute.
4. Move upwards until a *yes* or *no* is found.

This gives a great flexibility. You can change the default file attribute on a group, class or at the default level, and affect thousand of files with one click. If you need to give special permissions to certain paths, simply use the yes or no state on those paths.

If you kind of feel lost after a while and wander what permissions the users actually has on what directories, use the Dir Access report to get a full overview.

The paths

The War FTP Daemon will not give the user access to any file or directory unless paths are defined. The first time the system is started it creates two paths on the default level:

1. \ Access to all files, with LIST only permission.

2. .\ The Server CWD with all access denied.

The first thing you should do when you start to add users is to delete the \ directory and set up your own directory permissions.

Unlike the Virtual File System path, you can use overlapping paths here. You can set up C:\ with LIST only permission, C:\pub with read permission and C:\pub\incoming with write only permission. In this case the user will be able to see all the files on you C: drive (except the files in C.\pub\incoming), download anything in the C.\pub hierarchy, except files in the C:\pub\incoming, where he only can upload files.

If you use the Virtual File System, the paths must use the Virtual File System path type (\path). Else you must use DOS type paths (C:\). When you start the Virtual File System all paths will be transformed to legal paths within the Virtual File System, or disabled. When you stop the Virtual file System all paths are transformed back to DOS paths.

To add a path, simply press the Add button and use the browser to find the directory you want. You can also type in an existing or unexistant path.

To delete a path, select the path you want to delete and press the Delete button.

Attributes

Files

Read. Gives the user permission to download files in this path.

Write. Gives the user permission to upload files in this path. *Note: Unless the **Delete** attribute is given, the* user *will **not** be permitted to replace existent files.*

Delete. Gives the user permission to delete or rewrite/append files in this path.

Execute. Gives the user permission to execute .bat and .exe files in this path (*not yet implemented*). *Note: This attribute has nothing in common with the Virtual File System UNIX execute flag for directories.*

Directories

List. Gives the user permission to list the contents of directories in this path.

Create. Gives the user permission to create directories in this path. This option will be disabled unless the *Recursive* permission is set.

Remove. Gives the user permission to remove (delete) empty directories in the path. This option will be disabled unless the *Recursive* permission is set.

Special

DENY. Denies the user access to do anything in this path.

Root. Specifies that this is the users root path. When the server needs to find the users root path, it will scan the users file access paths and look for this flag. If no root dir is found, it will scan the group, class and default levels. If no path is found it will fall back to use \ as root directory. If the \

path is undefined, or if the user does not have access to \, he will be denied access when he tries to log in. Note: When you set this flag, any other root flag on another directory in the current list will be cleared. *A* user *will not be able to move to a directory at a level above his root. If C:\pub\ftp is defined as root, the* user *will be denied access to C.\pub or anything else above C:\pub\ftp.* See the Root and Home dir for more information.

Home. Specifies the users home directory (the directory that will be the current one when he logs in). The server resolves this in the same manner as it resolves the Root directory. See the Root and Home dir for more information

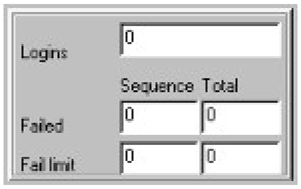
Mapping. This flag has two meanings:

> **On root paths**: Do not show the path down to the root path. If C:\pub\ftp is root, remove "C.\pub\ftp" and replace it with \ (or /). Internally the server will of course use the full path, but all paths transmitted between the server and the users FTP client will be transformed.
>
> **Non root paths**. Let the alias name of the path appear as a directory in the users root path. This option makes it convenient to create a customized file system for the user. *Note: If you define a path outside the* users root path*, you* ***must*** *map it. If it is not mapped, the* user *will not be able to see or access the path!*

Recursive. This flag tells the server to let the permissions set on this path apply for all it's sub-directories (all the way down). *You will usually use this flag on all the paths you define. If not, the permissions will only apply for the actual directory and not anywhere else.*

Login counter and the fail limit

Dialog: Security tab History



Logins 0

	Sequence	Total
Failed	0	0
Fail limit	0	0

Login counter

All successful logins are counted. This fields is provided for your information. If a user gives a bad password, it will affect the failed login counters.

Fail Limit

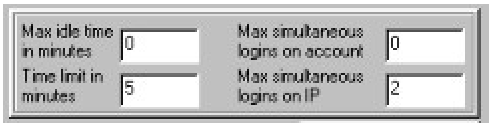
You can specify a limit on how many bad login attempts you accept from an account.

Fail limit sequence. The number of times a user can give a bad password before the server closes the account.

Fail limit total. The number of times a user can give a bad password before the user account is disabled. If you want to enable a user account that is disabled due to this, you must unchecked the disable account checkbox (or set it to undefined) *and* reset the total fail count to 0.

Idle time and Max simultaneous logins

Dialog: Security tab History



Max idle time in minutes 0

Max simultaneous logins on account 0

Time limit in minutes 5

Max simultaneous logins on IP 2

Idle time

The idle time is the time difference between the last request/data transmission to or from the user and the current time. It is common practice for FTP servers to end sessions when the idle time becomes to big, in order to serve more active users. In some cases the network connection can be lost without any TCP notification, and in these cases the only way to clear the user handle is to wait for the idle time time-out.

You specify the idle time as a number of minutes. If you use 0, the server will look for a defined idle time on a higher level. (If the user has idle time 0, and the class he belongs to has idle time 5 minutes, the user will time out after 5 minutes).

Time Limit

The time limit is the total time an account is allowed to remain logged in to the system. When the time limit is expired, the user will be logged out. If a file transfer is in progress, the logout will come right after the transfer is completed.

You specify the time limit as a number of minutes. If you use 0, the server will look for a defined time limit on a higher level. (If the user has idle time 0, and the class he belongs to has a time limit of 30 minutes, the user will time out after 30 minutes).

Max simultaneous logins

Some users, like the *anonymous* account, is shared among many persons. You can limit the number of simultaneous sessions for any account. If you leave this option to 0, the server will look for a defined limit at a higher level. If no limit is defined at any level, 0 will be interpreted as unlimited.

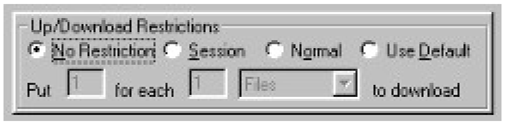
Max simultaneous logins on IP

This option allows you to restrict the number of simultaneous logins an account can have on one IP number. It is typically is used to restrict the sessions of anonymous users. You might allow 30 anonymous users, but only 1 or 2 connections from the same physical user. If you leave this option to 0, the server will look for a defined limit at a higher level. If no limit is defined at any level, 0 will be interpreted as unlimited.

The option above (Max simultaneous logins) take precedence over this one. So if you say 5 simultaneous logins and 10 simultaneous logins on IP, only 5 simultaneous logins will be allowed.

Up/Download Restrictions

Dialog: Security tab Security



Up/Download Restrictions

● No Restriction ○ Session ○ Normal ○ Use Default

Put 1 for each 1 Files to download

The Up/Download restriction feature is typically used on sites that want something back for the files they offer to the public. If enabled, the user will have to upload # files or bytes in order to download # files or bytes. If he don't give something, he don't get something either. In combination with the dupe checker this feature is very useful if you collect some sort of files.

If you have the Virtual File System running, files with the Free Download flag can be downloaded regardless of the modes described below.

Modes

No restriction. The option is disabled. This is how most FTP servers behave.

Session. The system compares the users upload/download counters for this session (since he logged on) and decides if he can download without first uploading something. This is the recommended class for all anonymous users on systems that require users to contribute before they get something back.

Normal. The system compares the users total upload/download counters to decide if he can download without first uploading something. This class should normally not be used for anonymous user accounts.

Use default. The system looks at the next level for the Up/Download restrictions.

Ratio

Put n for each n [Files/Bytes] to download. This is the definition of the up/download ratio. You can say upload 3 files/bytes for each 2 you get, or upload 1 file/byte for each 5 you get etc.

For simplicity, use small number. Do not say 100.000 bytes for each 200.000 bytes to download. Say 1 for each 2 bytes. The server does not care about the actual numbers, it looks at the relation between them when it decides if a user can download or not.

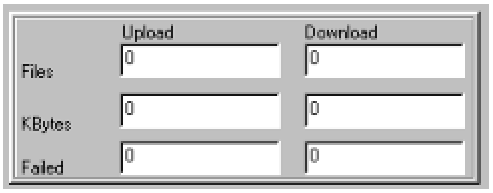
If you use *Bytes* as the rule, the server will look at the size of any requested file and deny download even if the user has uploaded something. If a user uploads a file on 50 Kb and the ratio is set to 1 : 2, he can download 100 Kb. If he request a file on 101 Kb, the request will be denied.

If you use *Files* as the rule, the server don't mind about the size. The user can upload a file of 1 KB and download 2 files of unlimited size (if the ratio is 1 : 2).

Note: When the Up/Download restrictions are active, a user *can not download anything (except free files) until he has made an upload.*

Up and Download Counters

Dialog: Security tab History



	Upload	Download
Files	0	0
KBytes	0	0
Failed	0	0

The War FTP Daemon remembers all regular file transfers. Both the number of files and the kilobytes transmitted are saved. Free files, directory listings, .Index.txt and .SysIdx.txt are not counted. The upload and download counters are used by the Up/Download restriction module.

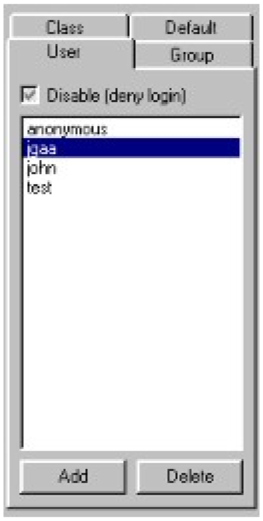
Aborted transfers are counted in separate counters.

You can edit these numbers.

Note: The statistics information uses it's own counters and are not affected by changes in these properties.

The User tab

Dialog: Security tab



If you accessed the Security tab via the Edit All command, you can select the level you want to alter in this tab. Else, you will only see one selection.

Enable or disable a user account

In order to enable or disable a user account, simply check or uncheck the ☑ Disable (deny login) checkbox. If you want to do this on a higher level, leave the box grayed (as on the sample). If all the boxes on all accounts are set to default (grayed), the checkbox on the system default properties can enable or disable all the accounts. If the checkbox on the user level is either checked or unchecked, this will override any settings on a highest layer for that user.

You can examine the access permissions by viewing the access report.

Adding and deleting users

To add a user, group or class, simply press the Add button. You will be asked for a user name and a password for the new user. Select a name and a password, or cancel the password dialog if you want the user account to use the email address as password.

If you add a group or a class, you will not be asked about password.

When you have added a new account, you can configure it, or leave it to use the default configuration. The default configuration for a user is the properties for the group, class and the default system properties added together.

A new user will be initiated to use the "ftp" group and the class defined in the default system properties.

A new group will be initiated to use the class defined in the default system properties.

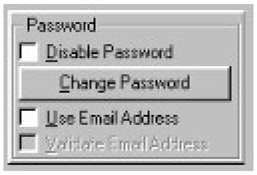
To make this a little clearer: Both the user, group and default level has the possibility to define a class they belong to. If no class is defined on the user level, the user will be assigned to the class defined on the group level. If the users-group don't have a class defined, the user will be assigned to the class defined on the default level. The default level will always be assigned to a class. So either way of the other - the user will always belong in a class. If you are uncertain about what users belong in which classes, you can press the Root/Home button for a list of all the users and the groups and classes they belong in.

To delete a user, select the user you want to delete and press the Delete button.

Note: You can not delete users, groups or classes created by the system. The system depends on being able to find the "Anonymous" user, the "ftp" group and the "Sysadmin" and "Visitor" classes. If you select one of these accounts, the delete button will be disabled.

Password

Dialog: Security tab Security



When a user is created he is either given a password, or (if you cancel the password dialog) will be asked to use his email address as password. You can change the users password or password properties at any time.

Password properties

Disable password. If checked, the server will not ask for password at all, but just accept the user once the user name is given. This can be useful for users that usually will come in via WEB browsers, as the browsers usually will give a fake email-address anyway. If the server don't ask for password, the login is faster, and the end-user more happy. *Note:* Users *without passwords are regarded as anonymous* users *by the properties entries on the System Console*.

Change password. This will open a dialog where you have to enter the new password twice. If the two passwords are equal, the new password is accepted. Users with real passwords are not regarded as anonymous users by the System Console. When the user log in, he will have to give the correct password. *Note: Passwords in the War FTP Daemon are case insensitive*.

Use Email Address. This is the common password mode for anonymous FTP sites. Many users will give fake addresses, but if you have 10 users on the same account logged inn simultaneously, it is easier to distinguish between them if they have individual email addresses. Even if they are fake.

Validate Email Address. This option is only significant if the option above is enabled. If enabled, the server will have a brief look at the users email address and verify that the formal email address syntax is correct. It will not take any action to verify if the email address exist. This option is first of all a way to prevent people from just entering "test" or "dickhead" as password.

The password is hidden somewhere in the encrypted user database and can not be easily broken. The security is far better than typical UNIX systems or other systems using the UNIX one-way encryption algorithm.

See also the Password Properties Report.

Access Report

Dialog: Security tab Access
Menu: View/Reports/User Access Privileges

The access report is a easy way to check the access (login) permissions for all the users on the system.

Sample report

```
User            Group           Class            S U G C D  =
--------------- --------------- ---------------  - - - - -  -
anonymous       ftp             Visitor          N     Y    N
jgaa            root            Sysadmin                    Y
john            ftp             User             N     N    N
smith           ftp             Guest                  Y    Y
test            ftp             Guest            N N   Y    N
```

Levels

S = System Console
U = User
G = user-group
C = User-class
D = Default

The SUGCD rows tells us about the permissions set on various levels, and the = row tells us if the user is allowed to log in or not.

Let's look at the sample report

anonymous is denied at the System Console. ☑ No anonymous logins is checked, and the server regards anyone without a password (or with the email address as password) as anonymous. The System console settings takes precedence over all other properties. So anonymous is denied access, even if he is allowed access on the Class level.

jgaa has not gotten any account access properties on any level and is granted access. The system will allow access in this situation.

john is denied on both class and System Console level and is denied access.

smith is allowed on the class level and is given access.

test is allowed on the class level, but denied at both System Console and User level and is denied access.

What you should avoid to do

Some system administrators sets the access properties on every box they can find. This is silly, as the design is made to permit allow or denial of many users by permitting or denying their group or class. If your report look like the sample below, you better read this chapter again and start to uncheck boxes :-)

```
User            Group           Class            S U G C D  =
--------------- --------------- ---------------  - - - - -  -
anonymous       ftp             Visitor          N N N Y Y  N
jgaa            root            Sysadmin           Y N N N  Y
```

```
john            ftp             User            N Y Y N Y  N
```

New users

If you create a new user and then execute the report before the system has updated the database, the line can look somewhat suspect.

```
User            Group           Class            S U G C D  =
--------------- --------------- ---------------  - - - - -  -
new user        [unassigned]    [unassigned]         - - -  Y
```

This is however quite normal. At this point the report claims that the user can log in and that the user does not belong in any group or class. The situation is that some internal tables are un-initialized, and the report generator is unable to resolve the information. The login module on the other hand, will not be able to find a user before the user database is updated and all the internal tables are updated. So this user will not be able to log in. Not yet.

If you press the update button the user account will be updated.

Users Home and Root dir Report

Dialog: Security tab Root/Home
Menu: View/Reports/User Home And Root Dir

```
User            Group           Class            Root dir.            Home dir.
--------------- --------------- ---------------  -------------------- --------------------
anonymous       ftp             Visitor          \httpnt              /httpnt
jgaa            root            Sysadmin         \                    /
john            ftp             User             \                    /
new user        ftp             Guest            \                    /
smith           ftp             Guest            \                    /
test            ftp             Guest            \                    /
```

This report simply lists all the users, the group and class they belong to, and their Root and Home directories.

If you new to know the permissions or at what level the root and home directories was defined, use the more verbose User Path Report.

User Path Report

Dialog: Security tab Root/Home
Menu: View/Reports/User Home And Root Dir

```
USERS AND THEIR ASSIGNED PATHS


anonymous
 RH[User   ] \httpnt                : List Read Maps to rootdir + Applies for subdirs.
   [User   ] \cgi                   : List Read
   [Default] \ukjent                : List Read
   [Default] \                      : List Read
   [User   ]  [default permissions] : List Read

jgaa
  H[User   ] \cgi                   : Create Delete Execute List Read Remove Write
   [Default] \ukjent                : Create Delete Execute List Read Remove Write  + Applies
for subdirs.
 R [User   ] \                      : Create Delete Execute List Read Remove Write  + Applies
for subdirs.
   [User   ]  [default permissions] : Create Delete Execute List Read Remove Write  + Applies
for subdirs.

john
   [Default] \ukjent                : All access denied + Applies for subdirs.
 RH[Default] \                      : List Read Write  + Applies for subdirs.
   [User   ]  [default permissions] : List Read Write
```

This report produce a list of all the users, and for each user, a list of all the paths defined in the Security tab's File Access Permissions list. This report prints the mix of paths and permissions that the system actually use. The lists are made up by the Path definitions on all levels (user, group, class and default), and their real attributes (when ☑ attributes are resolved to the [default permissions] at a higher level).

The report tells at what level the path was defined and all the permissions on each path.

Password Properties Report

Menu: View/Reports/User Home And Root Dir

```
User            Login class     Password status
--------------- --------------- -------------------------
anonymous       Anonymous       Disabled
jgaa            Administrator   Password Required
john            Anonymous       Email
new user        Normal          Password Required
smith           Normal          Password Required
test            Anonymous       Email
```

Login classes

Anonymous. All users without password enabled.

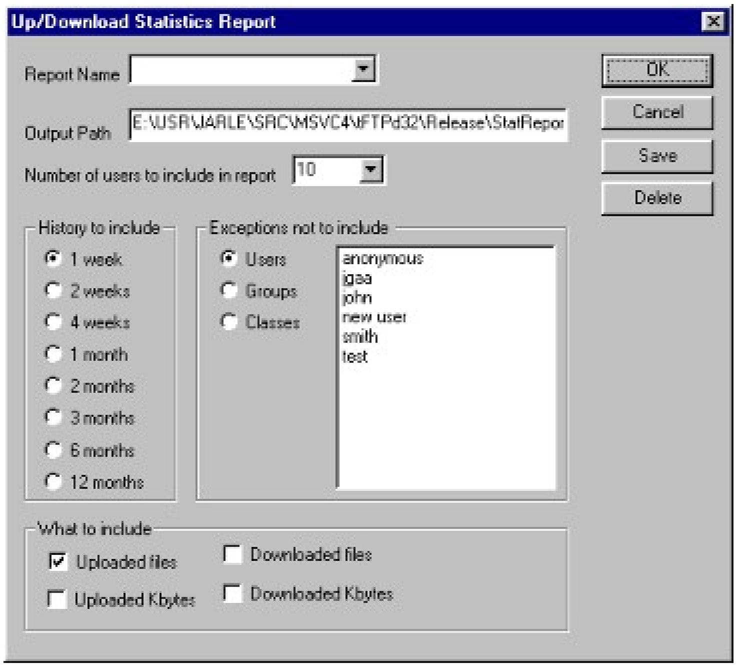
Normal. All regular users.

Administrator. All users in the class 'Sysadmin'

The *login classes* only applies when the user is logging on.

Up/Download Statistics Report (10 on top)

Menu: View/Reports/Up Download Statistics



Report Name

Output Path E:\USR\JARLE\SRC\MSVC4\FTPd32\Release\StatRepor

Number of users to include in report 10

History to include

- ● 1 week
- ○ 2 weeks
- ○ 4 weeks
- ○ 1 month
- ○ 2 months
- ○ 3 months
- ○ 6 months
- ○ 12 months

Exceptions not to include

- ● Users
- ○ Groups
- ○ Classes

anonymous
jgaa
john
new user
smith
test

What to include

- ☑ Uploaded files
- ☐ Uploaded Kbytes
- ☐ Downloaded files
- ☐ Downloaded Kbytes

From this dialog you can create sophisticated reports from the statistics data maintained on each user by the server.

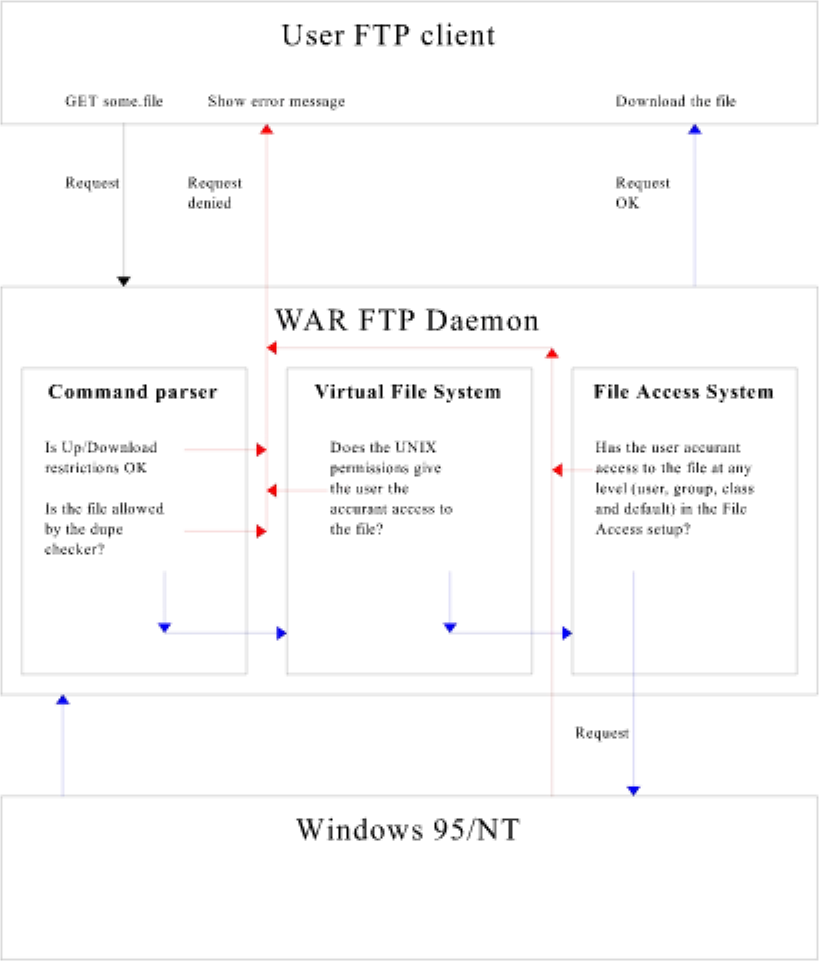
Creating a report

You can click on the option check and select-boxes, exclude users or groups of users and press the OK button to make an instant report.

If you will be running the same selections on a regular basis, you can give the report template a name and press the Save button.

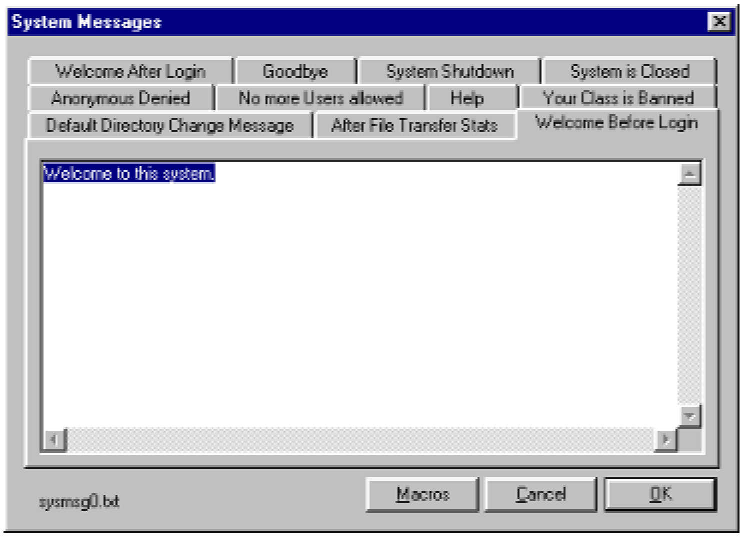
There is no limit in the number of report templates you can create.

File Access Security


User FTP client
GET some.file
Show error message
Download the file
Request
Request denied
Request OK
WAR FTP Daemon
Command parser
Is Up/Download restrictions OK
Is the file allowed by the dupe checker?
Virtual File System
Does the UNIX permissions give the user the accurant access to the file?
File Access System
Has the user accurant access to the file at any level (user, group, class and default) in the File Access setup?
Request
Windows 95/NT

System Messages

Menu: View/Messages



Welcome to this system.

sysmsg0.txt

The FTP specification states what commands the users FTP client can and shall send to the server, and how the server shall respond. Therefore is all the messages that is sent from the server to the users FTP client in response to different events hard-coded. However, in some cases it is useful to add some additional information. The War FTP Daemon can of course do this.

Just select the event that you want to make a little more verbose and write your message. You can of course use macros in all these messages.

Newline

The messages in the files are displayed via the standard [filename] macro. To allow a file to just contain a single word, and to avoid empty lines in the standard messages, no newline is inserted before the contents of the file. You will therefor normally begin your message with an empty line.

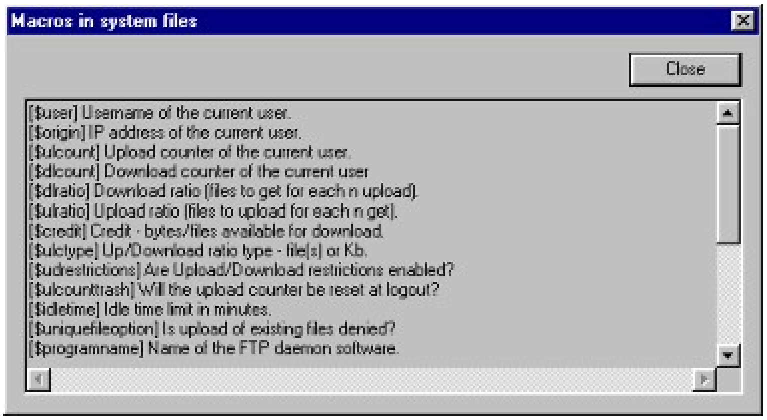
Altering the messages from other programs

All these messages are stored in textfiles in the Server CWD. You can modify these files from DOS .bat files or external programs (like adding special notes on holidays, when the backup program starts up, or have one message in the morning and another one in the afternoon).

The filename of the textfile is displayed at the left lower left corner of the dialog.

Macros

Menu: View/Messages
Dialog: System Messages Macros



[$user] Username of the current user.
[$origin] IP address of the current user.
[$ulcount] Upload counter of the current user.
[$dlcount] Download counter of the current user
[$dlratio] Download ratio (files to get for each n upload).
[$ulratio] Upload ratio (files to upload for each n get).
[$credit] Credit - bytes/files available for download.
[$ulctype] Up/Download ratio type - file(s) or Kb.
[$udrestrictions] Are Upload/Download restrictions enabled?
[$ulcounttrash] Will the upload counter be reset at logout?
[$idletime] Idle time limit in minutes.
[$uniquefileoption] Is upload of existing files denied?
[$programname] Name of the FTP daemon software.

When sending data from the server to the user in response to a command, the server looks for and expands macros.

Syntax of a macro

The formal syntax of a macro is [$tag] or [filename.ext]

The tags are hard coded in the server. For a complete list of macros, run the Macros dialog. The macros show up in a read-only text-window, allowing you to copy and paste the macros directly into where you need them.

Your own static messages as macros

As stated above, the tags are hard coded.

To make your own macro perform the following steps:

1. Create a textfile in the Server CWD.
2. Type in the message
3. Use the filename for this file as a macro in a greeting message or system message.

File-include macros can be recursive. If you call [mym1.txt] from [mym2.txt] and [mym2.txt] from [mym1.txt] the server will loop on this redundancy and *hang* until the user logs off. It is your responsibility to make sure that such situations does not occur.

Example

"Hi [$user]. Welcome to [$systemname]! Here is the latest news :-) [news.txt]"

```
230- Hi Joe. Welcome to Jgaa's Fan Club's FTP service! Here is the latest news :-)
230- --------------------------
230- bla bla
230- --------------------------
230 User logged in. Proceed.
```

How to show square brackets

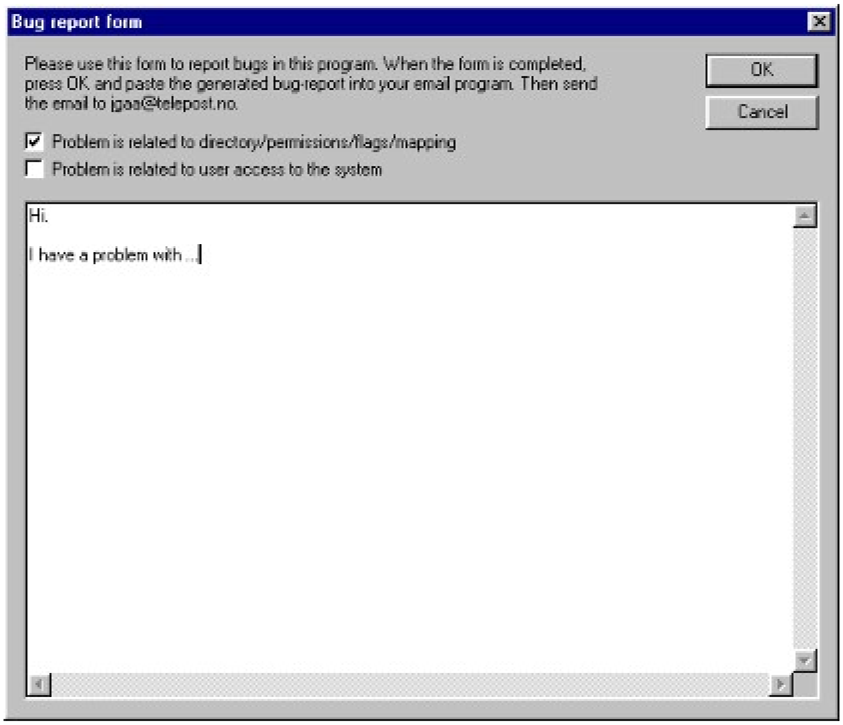
Sometimes you might want to use square brackets in your   messages. If you prefix the [bracket with backslash, the server will not try to expand a macro.

Hi \[well, this is just a test] whatever.

The] bracket can be used freely without any prefix.

Bug Reports

Menu: Help/Bug Report



Bug report form

Please use this form to report bugs in this program. When the form is completed, press OK and paste the generated bug-report into your email program. Then send the email to jgaa@telepost.no.

☑ Problem is related to directory/permissions/flags/mapping

□ Problem is related to user access to the system

Hi.

I have a problem with ...

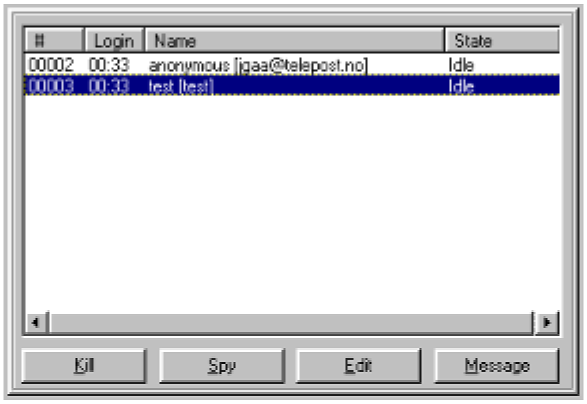
The War FTP Daemon is probably the most flexible and powerful FTP Daemon available for Windows today. This is great, but also has some disadvantages. As all advanced programs of this scale there are situations where the server will misbehave - simply because it is impossible to implement so much complexity and flexibility and then test all possible combinations of user setups, system setups, operating systems and patches, network cards and FTP clients.

Some problems can be present for a long period of time before someone come across a combination of settings and environment parameters that will trigger it. Other problems can occur when new features are implemented.

Bugfixes has a high priority for the developer. If you come across any problems, please use the Bug Report Form above and report it. If possible, supply detailed information about how the problem can be reproduced.

Thank you.

User List



#	Login	Name	State
00002	00:33	anonymous [jgaa@telepost.no]	Idle
00003	00:33	test [test]	Idle

The User List shows all the users currently online.

Rows

#. Handle number. This is a number that increments by one for each login from the server starts up. The number is used to distinguish between sessions if a user account is logged in several times.

Login. The time the user logged on.

Name. The name of the user. If the user has the Email password class, the password he typed will show up in square brackets.

State. The current state of the user.

Prelogin. The user is logging on.

Idle. The user is logged on, but is not doing anything.

Hold. The user has issued a command and are waiting for the server to respond.

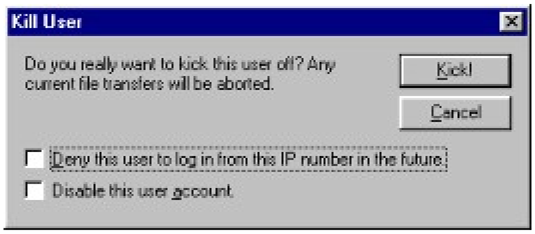
Upload. The user is uploading a file

Download. The user is downloading a file or a directory listing.

Buttons

Kill
Spy
Edit
Message

Kick (Kill) a user



Do you really want to kick this user off? Any current file transfers will be aborted.

□ Deny this user to log in from this IP number in the future.

□ Disable this user account.

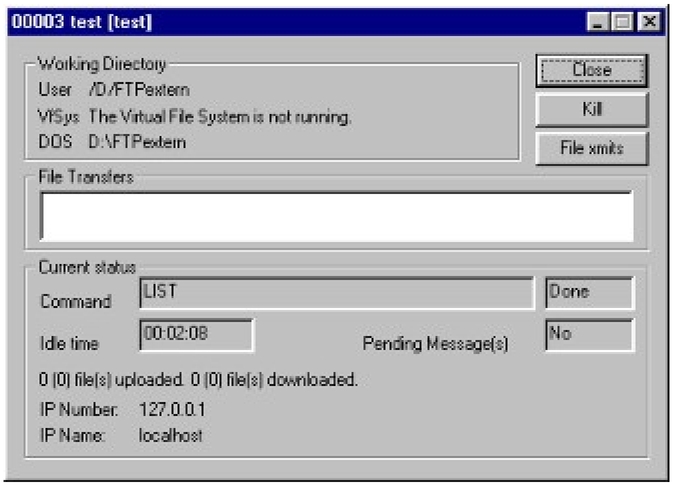
This command will allow you to end a user session.

Options

Deny this user... If checked, the IP number the user is using will be added to the IP Deny List for that user account. *Note: The denial applies for that* user *account only. The IP number will not be added to the default level IP Deny List.*

Disable this... If checked the user account will be disabled and any further login attempts denied.

Spy (on a user)



00003 test [test]

Working Directory

User /D/FTPextern

VfSys The Virtual File System is not running.

DOS D:\FTPextern

File Transfers

Current status

Command: LIST — Done

Idle time: 00:02:08 — Pending Message(s): No

0 (0) file(s) uploaded. 0 (0) file(s) downloaded.

IP Number: 127.0.0.1

IP Name: localhost

Working directory

User. The current path as the user knows it

VfSys. The current Virtual Vile System style path for the user.

DOS. The actual file system path.

File Transfers

If the user is uploading or downloading files the paths and progress will show up here.

Current status

Command. The last command issued by the user. *Note that some commands will map to other commands. The command name displayed is the actual command recognized and executed by the server*.

Idle time. The time difference between the last network message sent to or received from the user, and the current system time. If the user uploads or downloads a file, the idle time will be reset each time the server gets or sends data. The idle time is also updated when the user issues a command. If the idle time exceeds the Idle Time Limit specified in the Security tab, the user will be disconnected.

Pending message(s). Tells if the user has queued messages from you or from the system.

Statistics. File transfer statistics (number of files) uploaded and downloaded. The number in brackets are the counters for the current session.

IP Number. The IP number the user is calling from.

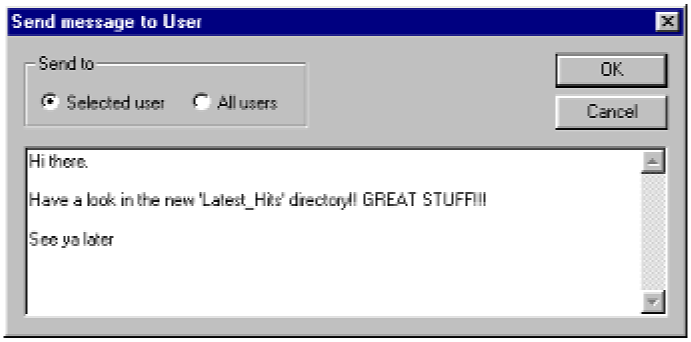
IP Name. When a user connects, the server issues a reverse DNS lookup on the user. If the name server returns a domain name, this will be displayed here.

Buttons

Kicks the user offline.

Shows a list of all regular files the user has uploaded or downloaded during the current session.

Send a Message to a user



Send to

● Selected user ○ All users

Hi there.

Have a look in the new 'Latest_Hits' directory!! GREAT STUFF!!!

See ya later

Send a message to one or all the users that are online.

FTP protocol considerations

The ability to send messages from the FTP server to the users FTP client was never thought of when the FTP protocol was designed. As a result there are no way to send asynchronous messages. And the FTP clients will not expect to receive this kind of messages.

The FTP protocol is very strict. The user sends a command and the server sends a definite or temporary answer. If the answer is temporary, the users FTP client will wait for a definite answer to follow.

This feature is implemented by queuing up messages and send them together with the next regular FTP protocol message to the user. If the user is transmitting a file, or examining a directory listing, it can take several minutes (or longer) before the message is sent. If the user just drops the connection the message will be trashed when the server cleans up the internal resources for that connection.

Another problem is that the message can disappear among other FTP protocol specific messages. And further more, some ftp clients will only show a few lines or no messages at all.

The conclusion is that this is a neat feature, but there is no guarantee that the message will be delivered. And if it is delivered, there is no guarantee that the user will see it.

Tip

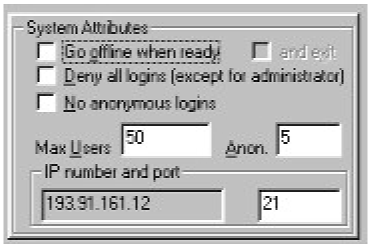
If you use this feature, make a multi line message with a little noise in it in order to get the attention of the user.

```
*************************************
*************************************

Hi there. How are you?

*************************************
*************************************
```

Messages as the one above is far easier to see than a message with only a line or two of text.

The users can also send a message to the System Console .

System Attributes



System Attributes

□ Go offline when ready □ and exit

□ Deny all logins (except for administrator)

□ No anonymous logins

Max Users 50 Anon. 5

IP number and port

193.91.161.12 21

System attributes

Go offline when ready. The server will go offline when the last user has ended his session. New logins will be refused unless the caller belongs in the *Sysadmin* class.

...and exit. When the server goes offline it will also shut down.

Deny all logins. All callers except those who belong in the *Sysadmin* class will be refused.

No anonymous logins. Only users with normal passwords will be allowed to log in.

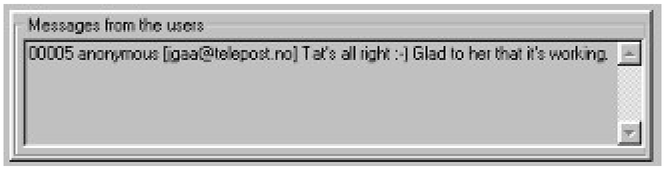
Max Users. The maximum number of concurrent sessions that is permitted. Users in the *Sysadmin* class can log in regardless of this setting. *Note: This number is the absolute total, including anonymous* users.

Anon. The maximum number of concurrent anonymous sessions that is permitted. Anonymous users are users without a normal password.

IP Number. The IP number on the current machine. On systems with several IP numbers it is undefined what IP number that will be reported. *Note: This number is only provided for your information. It is stored in the `.FtpDaemon.ini` file, but the server will never look at the number. Each time it goes online, it will ask the TCP/IP stack about the current IP number, and display the information returned.*

Port. Normally a FTP server will listen to port 21. Some sites use other numbers of security reasons, of because they run several FTP services on the same machine. You can enter the port number you want the server to use here. The new number will be user the next time the server goes online. See RFC 1700 for a list over port numbers and their assignments.

Messages from the users



00005 anonymous [jgaa@telepost.no] Tat's all right :-) Glad to her that it's working.

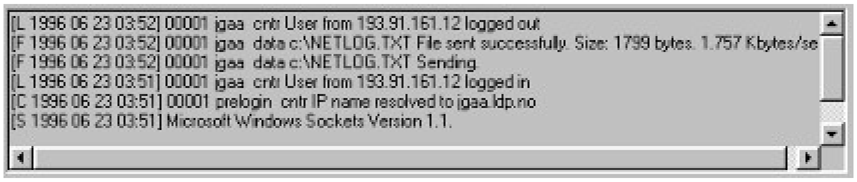
The FTP users can send messages to the System Console via the SITE MSG command. The messages will appear in the user Messages window in the order they come in.

The number of lines in this window is the same as the number of lines configured for the Log Window.

If you want a notification about incoming messages, you can let the server play a sound at this event.

You can also send a message back to the user.

Log Window



```
[L 1996 06 23 03:52] 00001 jgaa  cntr User from 193.91.161.12 logged out
[F 1996 06 23 03:52] 00001 jgaa  data c:\NETLOG.TXT File sent successfully. Size: 1799 bytes. 1.757 Kbytes/se
[F 1996 06 23 03:52] 00001 jgaa  data c:\NETLOG.TXT Sending.
[L 1996 06 23 03:51] 00001 jgaa  cntr User from 193.91.161.12 logged in
[C 1996 06 23 03:51] 00001 prelogin  cntr IP name resolved to jgaa.ldp.no
[S 1996 06 23 03:51] Microsoft Windows Sockets Version 1.1.
```

The log window displays the most recent logged events. You can specify which events you want to see and how many lines the window can contain in the Log Properties setup.

By default the log window will visualize the last messages. However, if you click on a line somewhere in the log, this line will remain the current selection, and the new messages will not be displayed before you scroll back to the top and click on the first line.

You can clear the log window.

Log

Like most server software, the War FTP Daemon has the ability to log a large number of events to logfiles. You can specify which events you want to see and how many lines the window can contain in the Log Properties setup. The most recent log evens will also be displayed in the Log Window on the System Console.

Log Format

The log is written to a standard DOS textfile where each line describe one log event.

```
[S 1996 06 23 03:51] WAR-FTPD 1.0b Copyright (c) 1996 by jgaa. WIN32 (WIN95)
[S 1996 06 23 03:51] Microsoft Windows Sockets Version 1.1.
[C 1996 06 23 03:51] 00001 prelogin  cntr IP name resolved to jgaa.ldp.no
[L 1996 06 23 03:51] 00001 jgaa  cntr User from 193.91.161.12 logged in
[F 1996 06 23 03:52] 00001 jgaa  data c:\NETLOG.TXT Sending.
[F 1996 06 23 03:52] 00001 jgaa  data c:\NETLOG.TXT File sent successfully. Size: 1799 bytes.
1.757 Kbytes/sec
[L 1996 06 23 03:52] 00001 jgaa  cntr User from 193.91.161.12 logged out
```

Log events

S: **System messages**.

L: Login/Logout. All events regarding logins and logouts.

F: File Access. Files uploaded, downloaded, deleted, renamed. Also directories that are created or removed.

C: Security messages. Users that are denied access, bad passwords etc.

W: System Warnings. Non-critical error situations in the different modules of the server.

E: Error messages. Critical error situations.

D: Debug messages. Generates a huge number of log output. Used to trace problems in the server or FTP clients.

Fields in the log

Event

Date and time when the event occurred

User **handle**

User **name**

Password (applies only for anonymous users)

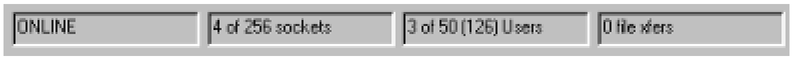
“cntr” or “data”: This tells if the event was reported from the control connection module or the data connection module. The control connection module handles all user commands and security, while the data connection module handles the file transmissions.

Filename: If the event regards a file transmission, a full DOS path is provided. Directory listings are reported as the path to the directory. Only regular files are listed.

Descriptive text: Some text to tell about the event.

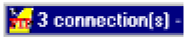
Note: Log events that does not origin from a user *connection will print the log event type, date and time, and then a descriptive text.*

System Status



ONLINE | 4 of 256 sockets | 3 of 50 (126) Users | 0 file xfers

The current system status is displayed on the bottom of the System Console.



3 connection(s) -

The Window Title is also updated with the current state (Offline, Idle or number of connections).

Running multiple FTP services on the same machine

You can run multiple FTP services on the same machine.

Port Numbers

Port 21 is defined as the FTP listening port. If you run multiple servers you must choose one port for each server. This is regardless of server software. See RFC 1700 for a list over port numbers and their assignments.

War FTP Daemon considerations

If you will run several instances of the War FTP Daemon on the same machine, follow the steps below.

1. Install War FTP Daemon and configure it to fit your needs. Do not run the Virtual File System.

2. Install a new copy of the War FTP Daemon in another directory. Do not use the same port number as any other services on your system, including other FTP servers. Do not use the Virtual File System.

3. Follow step 2 for each additional server you will use.

If the ftp servers are set up to use different parts of your hard-drives, and has no overlapping directories or files, you *can* run the Virtual file System.

When you run several servers, each server will have it's own user database and setup. The two servers will not share any information, except Directory Changes Messages spread around on your file system.

Multihoming

Multihoming is supported on the user level.

If you have several IP addresses assigned to your server you can make one "virtual" FTP server appear to each of the IP numbers by following the steps below:

Add a new anonymous user for each IP number. After the name, add @nnn.nnn.nnn.nnn, where nnn is the (server) IP number this user will log in to.

```
anonymous@127.0.0.2
anonymous@127.0.0.34
```

These users will only be allowed to log in when they connect to the IP number at the end of their name.

Do the same with any other users that are only allowed access via a certain IP address in the server.

Edit the sysmsg#.txt files and add the same suffix to their names.

```
sysmsg0@127.0.0.2.txt
sysmsg1@127.0.0.2.txt
```

If no `sysmsg` file is found for the IP number the user connects to, the default file will be displayed.

Note

The sysmsg files used for multihoming purpose can not be edited from the System Message dialog. You must edit these files with a text editor.

Users with no IP suffix in their name will be given access to any Server IP number. To deny the system defined anonymous account simply disable the account.

The IP suffix only works on the user level. It serves no purpose to add the suffix to the group or class names. You can however assign all users on a certain IP number to one group or class and use that to set up the default directory access etc.

If you use different server names on the different IP numbers, write the server name in the sysmsg#@ip.txt files and avoid the system name macro.

It is not possible to log on to the server giving a username with the IP suffix. The suffix is used by the system.

Multihoming using multiple servers

Version 1.65 of the War FTP Daemon introduce multihoming trough starting several instances of the server. To enable this feature, go to the Options/Server name dialog and specify what IP number the server should listen to. If no IP number is defined, the server will listen to all IP numbers assigned to the machine.

For each virtual domain, you must install one copy of the server into a dedicated directory, and set up the IP number it should use as described above.

Note: When you use several instances of the server, the user database is not shared.

Warning: You may not share the same physical directories/files among the instances of the server if you use the virtual file system! Doing so can and will destroy the security settings for VfSys, and will also most likely crash the servers! If you don't use VfSys, you may share the same up/download directories. The server itself must be installed in separate directories.

Specifications

Compatibility

- Follows the RFC 959 and 1123 FTP specifications.
- Works with Cute-FTP, WS_FTP, ftp (the one following Win95 and NT), MS Internet Explorer, Netscape Navigator, AmFTP, M-FTP, GuiFTP, - in fact, there is hardly reported any incompatibility at all.
- Emulates standard UNIX ftp servers, including most ls line parameters and ls formatted output.
- ABOR command supported. (Yes - it's listed in the RFC, but not all PC based FTP servers can handle it).
- REST command supported (Resume an aborted transfer)
- Import filter for Serv-U user database
- Native support for NT service process mode.
- Optional Wu-ftpd compatible log for file transfers
- Operates on Windows95, Windows NT 3.51 (Intel only) and Windows NT 4.0 (Intel only). War FTP Daemon is a native 32 bit application and will not work on Windows 3.*. Requires 486 processor or better.

Design

- Multithreading
- Dialog based system console
- Simple and advanced mode of operation, making it perfect for both small and large FTP services.
- Users are organized in user-groups and user-classes to ease maintenance of time-out, directory/file access etc.
- Automatic shutdown option when the last user has logged off
- Displays a list of all users online where you can kick one of them, edit the user database entry, send a message to the user or even spy at the user (see the last given command, current directory, file transfers and a list of all files this user has sent/received during the current session.
- Supports user supplied messages to the console through the SITE MSG command.
- Supports personal greeting messages to all users, groups and classes.
- Full OLE support for Win95/NT4 .lnk shortcuts
- Online and off-line operation
- In-memory transfer of directory listings
- DOS or UNIX style directory paths
- Mapping of paths to the root path
- Support for long filenames
- Virtual file system for improved performance and functionality, including full UNIX-like security and links on directory and file level.
- Scheduled re-scans of the Virtual File system to handle changes made on the file system outside the server.
- "On the fly" generation of index files for the current directory/all files on the system (that the user is allowed to see)
- Support for comments on any file/directory in "on the fly" generated .index files.
- Macros in the welcome files to display the users name, time-out limit etc.
- On single network/ppp connections, the server shows the current IP number in an edit window, allowing copy/paste of the number to other applications. It also saves this number to a file.
- Multihoming/multihosting support allows setup of several virtual FTP servers based on the IP number the user connect to.
- Context-sensitive Help
- Mounting of network drives at startup (NT service mode only)
- Denial of upload if the free disk space runs too low
- Flexible verification option on incoming files. This feature allows you to use “plug-in” modules (external programs) to verify and validate incoming files before the upload is accepted by the server. Can be used for CRC checking, virus scanning, processing of file_id.dix files etc. Several free plug-in modules are made available by experienced War FTP daemon users.

Security

- High level of security with directory-level permissions for read, write, execute, dir, mdkdir, rmdir and delete.
- Full UNIX security with user group and rwx permissions if the Virtual File System is used
- Upload/Download ratio option for anonymous and regular users. Supports file or byte check (upload # files/bytes for each # files/bytes to download). Free files can be defined.
- Reports of user access privileges, directory access privileges, etc.
- Maximum number of concurrent connections for users, groups and classes.
- Maximum number of connections based on login IP number (allows you to have ## anonymous logins, but just # concurrent connections from one workstation. Handy to prevent one user from taking up too many connections.
- Simple one-button enable or disable option for users, groups and classes
- IP level access control for the system, users, groups and classes. Incoming connections can be refused before and after login based on the callers IP address.
- Automatic disable of an account if a selectable number of bad passwords are given
- "Not Wanted" list of files that will be refused.
- "Dupe checker" that can deny upload of any filename that exist within the paths of the server
- Disconnect single users without shutting the system down
- No built-in limitations ("crippling"), "secret" connections to my site or any other ugly stuff.

Significant changes since 1.20b

- Several important bug-fixes
- The memory usage in the Virtual File System is reduced by 68%
- The Virtual File System is optimized. Load time is reduced ~70% and flush time reduced to ~ 10 - 100 ms.
- Many new features. NT users will especially appreciate the NT service mode.
- Improved performance
- Improved stability

Technical Support

How to get support

Updated support information and FAQ are available at:

USA: http://www.jgaa.com
Norway: http://home.sol.no/jgaa/

You can also use the usenet group news://alt.comp.jgaa and IRC Undernet #war_ftpd

Bug reports

Please use the Bug report Form if you think you have found a bug.

Files and file formats

Purpose	Name	Location	Format
User Database	FtpDaemon.dat	Server CWD	binary, encrypted
Setup	FtpDaemom.ini	Server CWD	text
System messages	sysmsg#.txt	Server CWD	text
Dir change msgs	.message.ftp.txt	Directories	text
Dir change msgs	.message.ftp.txt	Server CWD	text
VfSys data	.Index.txt	Directories	binary text, hidden
Links	*.lnk	Directories	Win95/NT4 shortcut
Log	LogFile.txt	Server CWD	text
Wu-ftpd log	(optional)	Server CWD	text
IP number	CurrentIPNumber.txt	Server CWD	text

IP number file

The IP number file “CurrentIPNumber.txt” is updated each time the servers goes online and contains the machines IP number as reported by Winsock.

The purpose of this file is to allow other programs to post or email the IP number when the server is available.

Wu-ftpd log file

Each server entry is composed of a single line of the following form, with all fields being separated by spaces.

current-time transfer-time remote-host file-size filename transfer-type special-action-flag direction access-mode username service-name authentication-method authenticated-user-id

current-time is the current local time in the form "DDD MMM dd hh:mm:ss YYYY". Where DDD is the day of the week, MMM is the month, dd is the day of the month,
hh is the hour, mm is the minutes, ss is the seconds, and YYYY is the year.

transfer-time is the total time in seconds for the transfer.

remote-host is the remote host name.

file-size is the size of the transfered file in bytes.

filename is the name of the transfered file.

transfer-type is a single character indicating the type of transfer. Can be one of:

a for an ascii transfer
b for a binary transfer

special-action-flag is one or more single character flags indicating any special action taken.
Can be one or more of:

C file was compressed (not used by War)
U file was uncompressed (not used by War)
T file was tar'ed (not user by War)
_ no action was taken (not user by War)

direction is the direction of the transfer. Can be one of:

o outgoing
i incoming

access-mode is the method by which the user is logged in. Can be one of:

(anonymous) is for an anonymous guest user.
(guest) is for an passworded guest user
(real) is for a local authenticated user.

username is the local username, or if guest, the ID string given.

service-name is the name of the service being invoked, usually FTP.

authentication-method is the method of authentication used. Can be one of:

none
RFC931 Authentication (not used by War)

authenticated-user-id is the user id returned by the authentication method. A * is used if an authenticated user id is not available. (War will currently always use * since it do not authenticate the users).

User Database

The user database is an encrypted binary file that contain all information about users, groups, classes and the default system properties. It is read when the server starts up and flushed to disk in 10 minutes intervals. It is also flushed to the disk when you exit the Security tab and when the server shuts down.

Location: Server CWD

.Index.txt (Physical file)

The Virtual File System flushes it's data to disk every 7 minutes. The information regarding a file is stored in the same directory as the file in a hidden file named .Index.txt. This makes it possible to move directories without loosing the extra information maintained by the Virtual File System.

The file is basically a normal textfile, and it can be edited in Notepad. The physical file is not accessible for users. The .Index.txt file the user is presented for is a special form of a directory listing.

```
WAR-FTPD 1.0b 3
^Aftp^A 0777 ^A(null)^A ^A(null)^A 5 0 ^Amain FTP dir^A
^Atmp^A 0777 ^A(null)^A ^A(null)^A 1 0 ^A(null)^A
*^AORO.ZIP^A 0666 ^ATommy^A ^A(null)^A ^Ad:\ftp\ORO.ZIP^A
^ABAL_GRE.GIF^A 0646 ^Aanonymous^A ^AVisitor^A 0 0 ^A(null)^A
^ABAL_GRY.GIF^A 0646 ^Aanonymous^A ^AVisitor^A 0 0 ^A(null)^A
^ABAL_PUR.GIF^A 0646 ^Aanonymous^A ^AVisitor^A 0 0 ^A(null)^A
^ABAL_RED.GIF^A 0646 ^Aanonymous^A ^AVisitor^A 0 0 ^A(null)^A
```

The first line identifies the file. The number after the version number of the server is the version number of the file. If you plan to write programs to manipulate these files, make sure not to manipulate files with version numbers higher than the one's you have documentation for.

Note that strings are enclosed by ^A (ASCII 1).

Older version of this file format

In version 2 of the .Index.txt file format, User names and Class names was not enclosed by `^A`.

C sample code

The code below is the server functions that reads and parses the file

```
// Perms manifests

#define NODE_OREAD              S_IRUSR // Owner read
#define NODE_0WRITE             S_IWUSR // Owner write
#define NODE_OEXEC              S_IXUSR // Owner execute

#define NODE_GREAD              S_IRGRP // Group read
#define NODE_GWRITE             S_IWGRP // Group write
#define NODE_GEXEC              S_IXGRP // Group execute

#define NODE_AREAD              S_IROTH // All read
#define NODE_AWRITE             S_IWOTH // All write
#define NODE_AEXEC              S_IXOTH // All execute

#define NODE_READ               0x0001  // Request read access
#define NODE_WRITE              0x0002  // Request write/delete access
#define NODE_EXEC               0x0004  // Request execute/chdir access
#define NODE_ANY                0x0008  // Request any access

#define NODE_LINK               0x1000  // This is a link to the file
#define NODE_LINK_O             0x8000  // This is also a .lnk link
#define NODE_DIRTY              0x2000  // Need to be flushed
#define NODE_KILLED             0x4000  // File is deleted

#define NODE_DEFAULT_FILE 0666
#define NODE_DEFAULT_DIR 0777

// Flags manifetst

#define INODE_DIR      0x0001  // Directory
```

```
#define INODE_DUPE     0x0002  // Allow duplicates
#define INODE_VIRTUAL  0x0004  // No dir, just a name
#define INODE_FREEDL   0x0008  // Free download

void CvfSys::LoadLevel(CvfNode *StartNode, LPCSTR Path)
{
  LOCK_VFSYS
  // Scan for .lnk files

  // Load info from the text file
  FILE *fp;
  CvfNode *Node, *LinkNode;

  if ((fp = fopen(Path,"r")) != NULL)
  {
    char *LineBuf = new char[2048];
    char *FileName = new char[MAX_PATH];
    char *RealPath = new char[MAX_PATH];
    char *Comment = new char[256];
    char *UserName = new char[64];
    char *ClassName = new char[64];
    int Perms, Flags, DlCnt;

    if (fgets(LineBuf,2047,fp) == NULL)
      goto done;

    int FileVersion;
    FileVersion = 0;
    sscanf(LineBuf,"%s %s %d", FileName,RealPath,&FileVersion);
    if ((FileVersion > 3) || (FileVersion < 2))
    {
      Log->LogMsg(LOGF_DEBUG,"CvfSys::LoadLevel(): Unknown .Index.txt file format.");
      goto done;
    }

    while(fgets(LineBuf,2047,fp) != NULL)
    {
      Perms = 0; Flags = 0;  *FileName = 0; *RealPath = 0; *Comment = 0; *UserName = 0;
*ClassName = 0; DlCnt = 0;
      if (*LineBuf == '*')
      {
        // Link
         if (FileVersion == 2)
           MySscanf(LineBuf+1,"\1%s\1 %O %s %s \1%s\1",
             FileName, &Perms, UserName, ClassName, RealPath);
         else
           MySscanf(LineBuf+1,"\1%s\1 %O \1%s\1 \1%s\1 \1%s\1",
             FileName, &Perms, UserName, ClassName, RealPath);

        if (*UserName == '(') *UserName = 0;
        if (*ClassName == '(') *ClassName = 0;
        if ((Perms == 0) || !*RealPath || (*RealPath == '('))
          continue; // Bad data...
        if ((LinkNode = ResolveNodeFromDosPath(RealPath)) == NULL)
          continue; // Faled to find link node
        Node = AddLink(LinkNode,StartNode->m_Father);
        Node->m_Perms &= ~0xfff;
        Node->m_Perms |= (Perms & 0xfff) | NODE_LINK ;
        if (*UserName) Node->m_User = strdup(UserName);
        if (*ClassName) Node->m_Class = strdup(ClassName);
      }
      else
      {
         // Normal file info
         if (FileVersion == 2)
           MySscanf(LineBuf,"\1%s\1 %O %s %s %d %ld \1%s\1",
             FileName, &Perms, UserName, ClassName, &Flags, &DlCnt, Comment);
         else
           MySscanf(LineBuf,"\1%s\1 %O \1%s\1 \1%s\1 %d %ld \1%s\1",
             FileName, &Perms, UserName, ClassName, &Flags, &DlCnt, Comment);
        if (*UserName == '(') *UserName = 0;
        if (*ClassName == '(') *ClassName = 0;
        if (!stricmp(Comment,"(null)")) *Comment = 0;
        if (!*FileName)
          continue; // Bad data...

        if ((Node = ResolveNodeFromName(FileName,StartNode)) == NULL)
        {
          // Try .lnk file
          int len = strlen(FileName);
          if ((len > 4) && !stricmp(FileName + (len - 4),".lnk"))
          {
            FileName[len - 4] = 0;
            if ((Node = ResolveNodeFromName(FileName,StartNode)) != NULL)
            {
              if (Node->m_Perms & NODE_LINK_O)
                goto go_on;
            }
          }
          continue; // The file must be deleted...
        }
go_on:
        Node->m_Perms &= ~0xfff;
        Node->m_Perms |= (Perms & 0xfff);
        if (*UserName)
        {
          if (Node->m_User)
            delete Node->m_User;
          Node->m_User = strdup(UserName);
        }
        if (*ClassName)
        {
          if (Node->m_Class)
            delete Node->m_Class;
          Node->m_Class = strdup(ClassName);
        }
        if (*Comment)
        {
          if (Node->m_Inode->m_Comment)
            delete Node->m_Inode->m_Comment;
          Node->m_Inode->m_Comment = strdup(Comment);
        }
        Node->m_Inode->m_Flags = Flags;
        Node->m_Inode->m_DlCnt = DlCnt;
      }
    }

done:
    delete ClassName;
    delete UserName;
    delete Comment;
    delete RealPath;
    delete FileName;
    delete LineBuf;
    fclose(fp);
  }

  // Second pass
  for(Node = StartNode; Node; Node = Node->m_Next)
  {
    if ((Node->m_Inode->m_Flags & INODE_DIR) && !(Node->m_Perms & NODE_LINK) && Node->m_Son)
      LoadExtraInfo(Node);
  }
}


int MySscanf(char *buf, LPCSTR Format, ...)
{
  va_list marker;
  va_start(marker, Format);
  int Rval = 0;
  char *p;
  int *i;
  int base;

  while(*Format)
  {
    if (*Format == '%')
    {
      switch(*++Format)
      {
        case '%':
          *buf++ = '%';
          break;
        case 's':
          p = va_arg(marker,char *);
          while(*buf && (!Format[1] || (*buf != Format[1])))
            *p++ = *buf++;
          *p = 0;
          break;
        case 'l':
          ++Format; // %ld, skip over one to get in sync
        case 'O': // Octal or decimal base.
          base = 10;
          i = va_arg(marker, int *);
          *i = 0;
          if (*buf == '0') // First digit is 0. Use octal
            base = 8; // Octal
          while(*buf && isdigit(*buf))
          {
            *i *= base;
            *i += *buf - '0';
            ++buf;
          }
          break;
        case 'd':
          base = 10;
          i = va_arg(marker, int *);
          *i = 0;
          while(*buf && isdigit(*buf))
          {
            *i *= base;
            *i += *buf - '0';
            ++buf;
          }
          break;
      }
      ++Format;
    }
    else
    {
      while(*buf && (*buf != *Format))
        ++buf;
      ++Format;
      ++buf;
    }
  }

  va_end(marker);
  return Rval;
}
```

.Index.txt and .SysIdx.txt (Virtual files)

If you use the Virtual File System and have the option enabled, these two files will show up in all directories.

If the user issues a download command, the command will be mapped to ls -I or ls \ -RI, and in stead of sending the requested file, the user will receive a special directory listing as it was a file.

This listing contains the filename, the number of times the file have been downloaded and a comment (if present).

Some FTP clients, like Cute-ftp, will look for a file named *index* and automatically download and display the comment after the filename. The download count will then appear as part of the comment.

If you want a more verbose listing of the files at your site, you can log in with the ftp client shipped with Win95/NT, and issue the following commands:

```
CD \
DIR -RI c:\temp\list.txt
```

This will produce a very verbose listing of all the files the server know about and save it as c:\temp\list.txt.

Links

A link is a filename that points to another file or directory. They are useful to ease navigation on a FTP system.

Supported link types

Directory mapping . Maps any directory to the users root path.

OLE Win95/NT .lnk **shortcut files**. If the Virtual File System is running it will transform these shortcuts to UNIX style links to speed up the OLE processing. You can use the Win95/NT4 Explorer to create and maintain shortcuts. The Virtual File System will remove the .lnk extortion from the links. If the Virtual File System is not running, the server will resolve the shortcut files as if they was links. But in this case the .lnk extension will be visible.

UNIX links. The Virtual File System can maintain links as if they were part of the file system. You create and delete such links via the View command.

If the virtual file system is running, you can assign comments and UNIX permissions to shortcut links and UNIX links.

The LIST command

The LIST command (issued by the users FTP client whenever the user want a directory listing) is the headache for all FTP client and server developers. There is no standardized format of the output.

The WAR FTP Daemon has it's own version of the UNIX *ls* command built in, and calls this when it receives a LIST command. The ls output is understood by all modern FTP clients and WEB browsers. The disadvantage is that the time and date shown is relative to the servers local time zone, and that the file time of older files is lost.

Formal syntax

LIST [<void>] | [pattern] | [arguments]

The ls module can handle *both* pattern and arguments, but this is currently not supported by the FTP protocol. The user must therefore CD to the directory if he want to use some of the arguments.

Arguments

```
-A      List all entries except for `.' and `..'.

-C      Force multi-column output

-F      Display a slash (/) immediately after each pathname that is a di-
        rectory, an asterisk (*) after each that is executable, and an at
        sign (@) after each symbolic link.

-L      If argument is a symbolic link, list the file or directory the
        link references rather than the link itself. See also the
        How To Display Links option.

-R      Recursively list subdirectories encountered.

-a      Include directory entries whose names begin with a dot (.).

-d      Directories are listed as plain files (not searched recursively)
        and symbolic links in the argument list are not indirected
        through.

-i      For each file, print the file's file serial number (inode num-
        ber).

-l      (The lowercase letter ``ell.'')  List in long format. (See be-
        low.)  If the output is to a terminal, a total sum for all the
        file sizes is output on a line before the long listing.

-1      (The numeric digit ``one.'')  Force output to be one entry per
        line.  This is the default when output is not to a terminal.

-I      Include header text, comments and download counter.

-S      Interpret the path as a single file or directory
```

Patterns

Ordinary pattern matching with * and ? is supported for the file name.

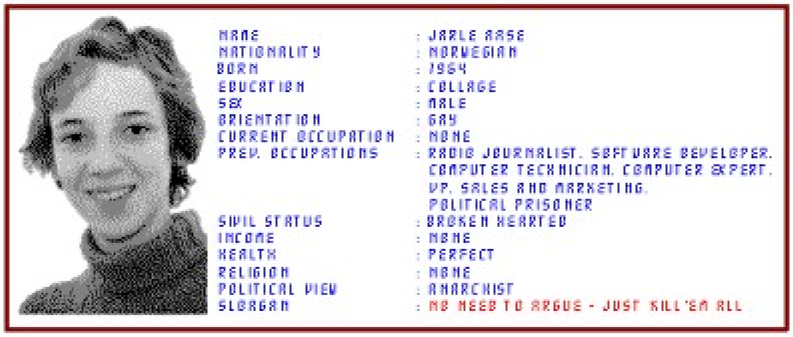
Limitations

Sorting and intelligent pattern matching (`/*/*.zip`) is not supported.

Tips

If you use the -L argument, links will be displayed regular files or directories without the 'l' flag.

About the author



NAME	: JARLE AASE
NATIONALITY	: NORWEGIAN
BORN	: 1964
EDUCATION	: COLLAGE
SEX	: MALE
ORIENTATION	: GAY
CURRENT OCCUPATION	: NONE
PREV. OCCUPATIONS	: RADIO JOURNALIST, SOFTWARE DEVELOPER, COMPUTER TECHNICIAN, COMPUTER EXPERT, VP, SALES AND MARKETING, POLITICAL PRISONER
SIVIL STATUS	: BROKEN HEARTED
INCOME	: NONE
HEALTH	: PERFECT
RELIGION	: NONE
POLITICAL VIEW	: ANARCHIST
SLOGAN	: NO NEED TO ARGUE - JUST KILL'EM ALL

Well... I don’t like to expose myself, but on the other hand, I do receive lot’s of emails from people that wonder who I am, and why I release my software as freeware. Of course, I also receive lot’s of “business opportunities” from people that realize the commercial potential of my programs and believe that I’m some kind of jerk they can rip off...

To realize why I do what I do, you have to know a little bit about the Norwegian society.

Norway is among the richest countries in the world. The country is well known for it’s efforts to make peace in the World, to secure the environment and to enforce human rights everywhere. The government is very eager to make the world a better place - at least *outside* the Norwegian borders.
Most people here believe that Norway is the best place on earth. I once did too. Until one fatal night about 12 years ago.

At that time I was among the most popular radio journalists in my hometown. One night I was arrested by the police and asked to lay against a person I did not now, in order to get him convicted for burglary. I refused. The next day the police wanted to imprison me, unless I signed their story. My councilor advised me to give a false testimony, sign the statement they wanted and walk out as a free man, as if this was the most natural thing in the world. I was shocked. But I refused and was imprisoned. One week later the police realized who I was, and set me free, with many excuses and hopes that I had “no hard feelings”.
When I got back behind the microphone I started to talk about police brutality, bestiality and injustice. About the same time other groups realized what was going on and the police got accused for brutality by a large number of victims. The “good citizens” of the town was shocked about these “wild” accusations, the press called them liars, and many of them was later convicted for “false accusations” against the police. Later on it is proven behind any doubt that the police in my hometown is brutal, that they indeed break the law, imprison innocent people if they feel like it and so on. Even Amnesty International has confirmed the situation. The government however refuse to do anything about it. It is a tradition in Norway that no one in the legal system is charged for any crimes they commit “in the line of duty”. A few years back a drunk police officer strangled a teenager, on his spare time, and got away with it.
I have seen and heard people being tortured in the basement of the local police station. Early 1996 a prisoner was burned to death in his cell in the local prison, after almost a year of illegal imprisonment. They called that suicide.

In 1992 I got into a dispute with one of the criminal police officers in the town. He threatened me several times on my life. To his surprise I got angry, not afraid. When a 12 year old friend of mine run away from an institution and hide in my apartment, while I was in another city, he decided to “get me”. I was arrested just after I returned home, but I kept my mouth shot about my alibi. The police thought that I had been in town the entire 2 weeks the boy was missing, and forced him to make false accusations about sexual abuse. Since I am gay, he believed that a conviction would come real easy. When they learned about my alibi they realized that they had a problem. In desperation they started to look for new accusations, and arrested a huge number of my friends in order to force them to testify against me - about anything. After 1 year they finally found a boy I once knew, who was mentally ill and accused me for murder, attempted murder and rape. The experts that examined the boy concluded that he was lying. His family believed he was lying, and so did his friends.
In court I managed to prove that the boy was lying about almost everything in his statement. I also proved that the

police had lied in court, had produced false evidence, had harassed my friends, and that I had an alibi for the charges about sexual abuse. The prosecutor did not even try to deny this. He attacked me for being gay (something that is perfectly legal in Norway), and asked the jury to convict me for my political opinions and my sexual orientation. And the jury followed his advice. They knew that I was innocent, and still answered yes to most of the questions, including rape, because I dared to use my constitutional rights of free speech, because I dared to be idealistic, and because I dared to criticize the conditions of the police and the legal system in Norway!

I was sentenced to 2 years in prison, and to pay a fine of about $8000 (a little more than I earned each month at that time). When the police 3 weeks later realized that I was working to get more hard evidences against them, the called out armed forces to get me. I was sent to jail 4 months before the scheduled time. Now I was real angry! I refused to talk with them, I refused to pay the fine (I actually sold, gave away or destroyed all my belongings to prevent the authorities to get a single cent from me). And in the prison I refused to work.

The prison I was sent to is the Norwegian “model prison”, a place that is presented to the press and the public as a very human place, with a great freedom for the prisoners, a place where “criminals” are turned into productive and well adjusted citizens. A very close friend of mine hung himself in that place in 1992. A prisoner that survived 5 years in on of the worst prisons in USA turned crazy after just 3 months of “special treatment”.

A lot happened during the 15 months I was kept as a political prisoner. To some of the staff it became an obsessions to “break” me. I was harassed, beaten, kicked, called “queer”, I found drugs planted in the room where I received visitors in a desperate attempt to frame me. I was threatened, refused food, money, soap and even toilet paper. I was kept isolated for a total of 8 months. Not once did they succeed in their attempts to manufacture new charges. Not once did they succeed in making me back off. My defense was passive resistance only. I never even raised my voice.

I am not a criminal. I am not a rapist. I don’t do drugs. I don’t even drink alcohol. But I am very, very angry. I refuse to accept a society that pretend to protect and encourage human rights and a better world, when the truth is that they prosecute human right activists and peaceful idealists with false, criminal charges.

My friends and my family has advised me to turn my back to the past and get back into a “normal life”. I can’t do that. My commitment to truth and justice is too strong, my sacrifices to these ideals too expensive. I have sworn that the Norwegian government shall walk over my dead body to get a single cent from me. I have sworn that the people that committed these crimes against me, and against the basic ideals of the population of this country; the police officers, the prosecutor, the jury, the judge - shall not get away with their crimes this time. Sometimes, very rarely, the criminals within the justice system choose the wrong person to mess with.

So, - why do I give away my software for free? There are several reasons. I don’t want to give the Norwegian government any money. Not for the fine, and not in taxes. Of course, I could sell the software aboard and take steps to prevent the money from falling into their hands. But that would be a crime. And as stated above - I am not a criminal. Not in any way! Another reason is my concern about free speech. Internet is being commercialized and censored, and the only way to secure the exchange of opinions , views, statements, and stories, is to make advanced communication software available to anyone. By providing the Internet community with free, high quality communications software, I hope that my contribution will help and encourage the exchange of all kind of information. Most of this information will be junk, some of it illegal, much of it also strongly against my personal opinions and values, - but that is the price we all have to pay if the free spirit shall survive.

And - why do I tell this story? Why do I expose my self for the risk of being called criminal, perverted or even rapist on the Internet? I do this because I have no other option. A false conviction is among the worst and most cruel crimes a person can be a victim of. I demand the right to be recognized as a victim of crimes, and not as an offender. I demand the right to be recognized as the person I am; a person with dignity, ideals and moral standards. You can choose to believe me, or you can choose to regard me as a liar - but frankly - would any sane person do what I do and tell a story like this, unless it is the truth?

The **server CWD** is normally the directory where the server is installed.

A **link** is a filename that points to another file. Links are usually used to ease navigation by providing aliased directories the user can CD to.

User: A person or a process on behalf of a person wishing to obtain file transfer service. The human user may interact directly with a server-FTP process, but use of a user-FTP process is preferred since the protocol design is weighted towards automata.

Group: A template with user properties. All users are assigned to a group, and all users inherits the shared properties of their group unless (expect for those properties overridden by the users own properties).

Multihoming: To let the server appear as several ftp services, depending on the IP number the user connects to. Require the server machine to have several IP numbers assign to the TCP/IP stack.

Levels: Many options can be set in the user, group, class and default properties. When the server needs to know the state of such an option, it scans the different 'levels' for a determinate answer. If it find a ☑ or □ it use that value. If the option is set to ▨ it looks at the next level. Most options are scanned in this order: user >> group >> class >> default.

Control connection: The network connection between the user FTP client and the server, where the FTP commands and reply codes are sent. This connection remains open during the entire session.

Data connection: A network connection created for file transfers and directory listings. The Data connection will usually be closed when the transfer is complete and a new connection created when the next transfer is about to start.

Root path: The entry point to the file system for a user.

This file will be sent to the user over the control connection when he enters the directory where it is located.

If present, this file will be sent to the user over the control connection when he enters a directory that does not have its own .message.ftp.txt file.

DES13, or just DES stands for *Data Encryption Standard*, and was the common scramble algorithm used to protect passwords in UNIX. Today there are several new and more secure algorithms, but DES is still widely used. Any key scrambled with DES13 will result in a random 13 bytes long string of text. Authentication programs, like UNIX login and the War FTP Daemon can determine if a typed password correspond with a scrambled password.

Internal

link

Server CWD

user

users

user-group

user-groups

multi-homing

level.message.ftp.txt*!popupId(ex_message_ftp)
.message.ftp.txt

data connection

control connection

root path

Internet Port Numbers (RFC 1700)

WELL KNOWN PORT NUMBERS

The Well Known Ports are controlled and assigned by the IANA and on most systems can only be used by system (or root) processes or by programs executed by privileged users.

Ports are used in the TCP [RFC793] to name the ends of logical connections which carry long term conversations. For the purpose of providing services to unknown callers, a service contact port is defined. This list specifies the port used by the server process as its contact port. The contact port is sometimes called the "well-known port".

To the extent possible, these same port assignments are used with the UDP [RFC768].

The assigned ports use a small portion of the possible port numbers. For many years the assigned ports were in the range 0-255. Recently, the range for assigned ports managed by the IANA has been expanded to the range 0-1023.

Port Assignments:

Keyword	Decimal	Description	References
	0/tcp	Reserved	
	0/udp	Reserved	
#		Jon Postel <postel@isi.edu>	
tcpmux	1/tcp	TCP Port Service Multiplexer	
tcpmux	1/udp	TCP Port Service Multiplexer	
#		Mark Lottor <MKL@nisc.sri.com>	
compressnet	2/tcp	Management Utility	
compressnet	2/udp	Management Utility	
compressnet	3/tcp	Compression Process	
compressnet	3/udp	Compression Process	
#		Bernie Volz <VOLZ@PROCESS.COM>	
#	4/tcp	Unassigned	
#	4/udp	Unassigned	
rje	5/tcp	Remote Job Entry	
rje	5/udp	Remote Job Entry	
#		Jon Postel <postel@isi.edu>	
#	6/tcp	Unassigned	
#	6/udp	Unassigned	
echo	7/tcp	Echo	
echo	7/udp	Echo	
#		Jon Postel <postel@isi.edu>	
#	8/tcp	Unassigned	
#	8/udp	Unassigned	
discard	9/tcp	Discard	
discard	9/udp	Discard	
#		Jon Postel <postel@isi.edu>	
#	10/tcp	Unassigned	
#	10/udp	Unassigned	
systat	11/tcp	Active Users	
systat	11/udp	Active Users	
#		Jon Postel <postel@isi.edu>	
#	12/tcp	Unassigned	
#	12/udp	Unassigned	
daytime	13/tcp	Daytime	
daytime	13/udp	Daytime	
#		Jon Postel <postel@isi.edu>	
#	14/tcp	Unassigned	
#	14/udp	Unassigned	
#	15/tcp	Unassigned [was netstat]	
#	15/udp	Unassigned	
#	16/tcp	Unassigned	

```
#                16/udp    Unassigned
qotd             17/tcp    Quote of the Day
qotd             17/udp    Quote of the Day
#                          Jon Postel <postel@isi.edu>
msp              18/tcp    Message Send Protocol
msp              18/udp    Message Send Protocol
#                          Rina Nethaniel <---none--->
chargen          19/tcp    Character Generator
chargen          19/udp    Character Generator
ftp-data         20/tcp    File Transfer [Default Data]
ftp-data         20/udp    File Transfer [Default Data]
ftp              21/tcp    File Transfer [Control]
ftp              21/udp    File Transfer [Control]
#                          Jon Postel <postel@isi.edu>
#                22/tcp    Unassigned
#                22/udp    Unassigned
telnet           23/tcp    Telnet
telnet           23/udp    Telnet
#                          Jon Postel <postel@isi.edu>
                 24/tcp    any private mail system
                 24/udp    any private mail system
#                          Rick Adam <rick@UUNET.UU.NET>
smtp             25/tcp    Simple Mail Transfer
smtp             25/udp    Simple Mail Transfer
#                          Jon Postel <postel@isi.edu>
#                26/tcp    Unassigned
#                26/udp    Unassigned
nsw-fe           27/tcp    NSW User System FE
nsw-fe           27/udp    NSW User System FE
#                          Robert Thomas <BThomas@F.BBN.COM>
#                28/tcp    Unassigned
#                28/udp    Unassigned
msg-icp          29/tcp    MSG ICP
msg-icp          29/udp    MSG ICP
#                          Robert Thomas <BThomas@F.BBN.COM>
#                30/tcp    Unassigned
#                30/udp    Unassigned
msg-auth         31/tcp    MSG Authentication
msg-auth         31/udp    MSG Authentication
#                          Robert Thomas <BThomas@F.BBN.COM>
#                32/tcp    Unassigned
#                32/udp    Unassigned
dsp              33/tcp    Display Support Protocol
dsp              33/udp    Display Support Protocol
#                          Ed Cain <cain@edn-unix.dca.mil>
#                34/tcp    Unassigned
#                34/udp    Unassigned
                 35/tcp    any private printer server
                 35/udp    any private printer server
#                          Jon Postel <postel@isi.edu>
#                36/tcp    Unassigned
#                36/udp    Unassigned
time             37/tcp    Time
time             37/udp    Time
#                          Jon Postel <postel@isi.edu>
rap              38/tcp    Route Access Protocol
rap              38/udp    Route Access Protocol
#                          Robert Ullmann <ariel@world.std.com>
rlp              39/tcp    Resource Location Protocol
rlp              39/udp    Resource Location Protocol
#                          Mike Accetta <MIKE.ACCETTA@CMU-CS-A.EDU>
#                40/tcp    Unassigned
#                40/udp    Unassigned
graphics         41/tcp    Graphics
graphics         41/udp    Graphics
nameserver       42/tcp    Host Name Server
nameserver       42/udp    Host Name Server
nicname          43/tcp    Who Is
nicname          43/udp    Who Is
mpm-flags        44/tcp    MPM FLAGS Protocol
mpm-flags        44/udp    MPM FLAGS Protocol
```

```
mpm              45/tcp    Message Processing Module [recv]
mpm              45/udp    Message Processing Module [recv]
mpm-snd          46/tcp    MPM [default send]
mpm-snd          46/udp    MPM [default send]
#                          Jon Postel <postel@isi.edu>
ni-ftp           47/tcp    NI FTP
ni-ftp           47/udp    NI FTP
#                          Steve Kille <S.Kille@isode.com>
auditd           48/tcp    Digital Audit Daemon
auditd           48/udp    Digital Audit Daemon
#                          Larry Scott <scott@zk3.dec.com>
login            49/tcp    Login Host Protocol
login            49/udp    Login Host Protocol
#                          Pieter Ditmars <pditmars@BBN.COM>
re-mail-ck       50/tcp    Remote Mail Checking Protocol
re-mail-ck       50/udp    Remote Mail Checking Protocol
#                          Steve Dorner <s-dorner@UIUC.EDU>
la-maint         51/tcp    IMP Logical Address Maintenance
la-maint         51/udp    IMP Logical Address Maintenance
#                          Andy Malis <malis_a@timeplex.com>
xns-time         52/tcp    XNS Time Protocol
xns-time         52/udp    XNS Time Protocol
#                          Susie Armstrong <Armstrong.wbst128@XEROX>
domain           53/tcp    Domain Name Server
domain           53/udp    Domain Name Server
#                          Paul Mockapetris <PVM@ISI.EDU>
xns-ch           54/tcp    XNS Clearinghouse
xns-ch           54/udp    XNS Clearinghouse
#                          Susie Armstrong <Armstrong.wbst128@XEROX>
isi-gl           55/tcp    ISI Graphics Language
isi-gl           55/udp    ISI Graphics Language
xns-auth         56/tcp    XNS Authentication
xns-auth         56/udp    XNS Authentication
#                          Susie Armstrong <Armstrong.wbst128@XEROX>
                 57/tcp    any private terminal access
                 57/udp    any private terminal access
#                          Jon Postel <postel@isi.edu>
xns-mail         58/tcp    XNS Mail
xns-mail         58/udp    XNS Mail
#                          Susie Armstrong <Armstrong.wbst128@XEROX>
                 59/tcp    any private file service
                 59/udp    any private file service
#                          Jon Postel <postel@isi.edu>
                 60/tcp    Unassigned
                 60/udp    Unassigned
ni-mail          61/tcp    NI MAIL
ni-mail          61/udp    NI MAIL
#                          Steve Kille <S.Kille@isode.com>
acas             62/tcp    ACA Services
acas             62/udp    ACA Services
#                          E. Wald <ewald@via.enet.dec.com>
#                63/tcp    Unassigned
#                63/udp    Unassigned
covia            64/tcp    Communications Integrator (CI)
covia            64/udp    Communications Integrator (CI)
#                          "Tundra" Tim Daneliuk
#                          <tundraix!tundra@clout.chi.il.us>
tacacs-ds        65/tcp    TACACS-Database Service
tacacs-ds        65/udp    TACACS-Database Service
#                          Kathy Huber <khuber@bbn.com>
sql*net          66/tcp    Oracle SQL*NET
sql*net          66/udp    Oracle SQL*NET
#                          Jack Haverty <jhaverty@ORACLE.COM>
bootps           67/tcp    Bootstrap Protocol Server
bootps           67/udp    Bootstrap Protocol Server
bootpc           68/tcp    Bootstrap Protocol Client
bootpc           68/udp    Bootstrap Protocol Client
#                          Bill Croft <Croft@SUMEX-AIM.STANFORD.EDU>
tftp             69/tcp    Trivial File Transfer
tftp             69/udp    Trivial File Transfer
#                          David Clark <ddc@LCS.MIT.EDU>
```

```
gopher           70/tcp    Gopher
gopher           70/udp    Gopher
#                          Mark McCahill <mpm@boombox.micro.umn.edu>
netrjs-1         71/tcp    Remote Job Service
netrjs-1         71/udp    Remote Job Service
netrjs-2         72/tcp    Remote Job Service
netrjs-2         72/udp    Remote Job Service
netrjs-3         73/tcp    Remote Job Service
netrjs-3         73/udp    Remote Job Service
netrjs-4         74/tcp    Remote Job Service
netrjs-4         74/udp    Remote Job Service
#                          Bob Braden <Braden@ISI.EDU>
                 75/tcp    any private dial out service
                 75/udp    any private dial out service
#                          Jon Postel <postel@isi.edu>
deos             76/tcp    Distributed External Object Store
deos             76/udp    Distributed External Object Store
#                          Robert Ullmann <ariel@world.std.com>
                 77/tcp    any private RJE service
                 77/udp    any private RJE service
#                          Jon Postel <postel@isi.edu>
vettcp           78/tcp    vettcp
vettcp           78/udp    vettcp
#                          Christopher Leong <leong@kolmod.mlo.dec.com>
finger           79/tcp    Finger
finger           79/udp    Finger
#                          David Zimmerman <dpz@RUTGERS.EDU>
www-http         80/tcp    World Wide Web HTTP
www-http         80/udp    World Wide Web HTTP
#                          Tim Berners-Lee <timbl@nxoc01.cern.ch>
hosts2-ns        81/tcp    HOSTS2 Name Server
hosts2-ns        81/udp    HOSTS2 Name Server
#                          Earl Killian <EAK@MORDOR.S1.GOV>
xfer             82/tcp    XFER Utility
xfer             82/udp    XFER Utility
#                          Thomas M. Smith <tmsmith@esc.syr.ge.com>
mit-ml-dev       83/tcp    MIT ML Device
mit-ml-dev       83/udp    MIT ML Device
#                          David Reed <--none--->
ctf              84/tcp    Common Trace Facility
ctf              84/udp    Common Trace Facility
#                          Hugh Thomas <thomas@oils.enet.dec.com>
mit-ml-dev       85/tcp    MIT ML Device
mit-ml-dev       85/udp    MIT ML Device
#                          David Reed <--none--->
mfcobol          86/tcp    Micro Focus Cobol
mfcobol          86/udp    Micro Focus Cobol
#                          Simon Edwards <--none--->
                 87/tcp    any private terminal link
                 87/udp    any private terminal link
#                          Jon Postel <postel@isi.edu>
kerberos         88/tcp    Kerberos
kerberos         88/udp    Kerberos
#                          B. Clifford Neuman <bcn@isi.edu>
su-mit-tg        89/tcp    SU/MIT Telnet Gateway
su-mit-tg        89/udp    SU/MIT Telnet Gateway
#                          Mark Crispin <MRC@PANDA.COM>
dnsix            90/tcp    DNSIX Securit Attribute Token Map
dnsix            90/udp    DNSIX Securit Attribute Token Map
#                          Charles Watt <watt@sware.com>
mit-dov          91/tcp    MIT Dover Spooler
mit-dov          91/udp    MIT Dover Spooler
#                          Eliot Moss <EBM@XX.LCS.MIT.EDU>
npp              92/tcp    Network Printing Protocol
npp              92/udp    Network Printing Protocol
#                          Louis Mamakos <louie@sayshell.umd.edu>
dcp              93/tcp    Device Control Protocol
dcp              93/udp    Device Control Protocol
#                          Daniel Tappan <Tappan@BBN.COM>
objcall          94/tcp    Tivoli Object Dispatcher
objcall          94/udp    Tivoli Object Dispatcher
```

```
#                          Tom Bereiter <--none--->
supdup           95/tcp    SUPDUP
supdup           95/udp    SUPDUP
#                          Mark Crispin <MRC@PANDA.COM>
dixie            96/tcp    DIXIE Protocol Specification
dixie            96/udp    DIXIE Protocol Specification
#                Tim Howes <Tim.Howes@terminator.cc.umich.edu>
swift-rvf        97/tcp    Swift Remote Vitural File Protocol
swift-rvf        97/udp    Swift Remote Vitural File Protocol
#                          Maurice R. Turcotte
#                <mailrus!uflorida!rm1!dnmrt%rmatl@uunet.UU.NET>
tacnews          98/tcp    TAC News
tacnews          98/udp    TAC News
#                          Jon Postel <postel@isi.edu>
metagram         99/tcp    Metagram Relay
metagram         99/udp    Metagram Relay
#                          Geoff Goodfellow <Geoff@FERNWOOD.MPK.CA.U>
newacct         100/tcp    [unauthorized use]
hostname        101/tcp    NIC Host Name Server
hostname        101/udp    NIC Host Name Server
#                          Jon Postel <postel@isi.edu>
iso-tsap        102/tcp    ISO-TSAP
iso-tsap        102/udp    ISO-TSAP
#                          Marshall Rose <mrose@dbc.mtview.ca.us>
gppitnp         103/tcp    Genesis Point-to-Point Trans Net
gppitnp         103/udp    Genesis Point-to-Point Trans Net
acr-nema        104/tcp    ACR-NEMA Digital Imag. & Comm. 300
acr-nema        104/udp    ACR-NEMA Digital Imag. & Comm. 300
#                          Patrick McNamee <--none--->
csnet-ns        105/tcp    Mailbox Name Nameserver
csnet-ns        105/udp    Mailbox Name Nameserver
#                          Marvin Solomon <solomon@CS.WISC.EDU>
3com-tsmux      106/tcp    3COM-TSMUX
3com-tsmux      106/udp    3COM-TSMUX
#                          Jeremy Siegel <jzs@NSD.3Com.COM>
rtelnet         107/tcp    Remote Telnet Service
rtelnet         107/udp    Remote Telnet Service
#                          Jon Postel <postel@isi.edu>
snagas          108/tcp    SNA Gateway Access Server
snagas          108/udp    SNA Gateway Access Server
#                          Kevin Murphy <murphy@sevens.lkg.dec.com>
pop2            109/tcp    Post Office Protocol - Version 2
pop2            109/udp    Post Office Protocol - Version 2
#                          Joyce K. Reynolds <jkrey@isi.edu>
pop3            110/tcp    Post Office Protocol - Version 3
pop3            110/udp    Post Office Protocol - Version 3
#                          Marshall Rose <mrose@dbc.mtview.ca.us>
sunrpc          111/tcp    SUN Remote Procedure Call
sunrpc          111/udp    SUN Remote Procedure Call
#                          Chuck McManis <cmcmanis@sun.com>
mcidas          112/tcp    McIDAS Data Transmission Protocol
mcidas          112/udp    McIDAS Data Transmission Protocol
#                          Glenn Davis <davis@unidata.ucar.edu>
auth            113/tcp    Authentication Service
auth            113/udp    Authentication Service
#                          Mike St. Johns <stjohns@arpa.mil>
audionews       114/tcp    Audio News Multicast
audionews       114/udp    Audio News Multicast
#                          Martin Forssen <maf@dtek.chalmers.se>
sftp            115/tcp    Simple File Transfer Protocol
sftp            115/udp    Simple File Transfer Protocol
#                          Mark Lottor <MKL@nisc.sri.com>
ansanotify      116/tcp    ANSA REX Notify
ansanotify      116/udp    ANSA REX Notify
#                          Nicola J. Howarth <njh@ansa.co.uk>
uucp-path       117/tcp    UUCP Path Service
uucp-path       117/udp    UUCP Path Service
sqlserv         118/tcp    SQL Services
sqlserv         118/udp    SQL Services
#                          Larry Barnes <barnes@broke.enet.dec.com>
nntp            119/tcp    Network News Transfer Protocol
nntp            119/udp    Network News Transfer Protocol
#                          Phil Lapsley <phil@UCBARPA.BERKELEY.EDU>
cfdptkt         120/tcp    CFDPTKT
cfdptkt         120/udp    CFDPTKT
#                          John Ioannidis <ji@close.cs.columbia.ed>
erpc            121/tcp    Encore Expedited Remote Pro.Call
erpc            121/udp    Encore Expedited Remote Pro.Call
#                          Jack O'Neil <---none--->
smakynet        122/tcp    SMAKYNET
smakynet        122/udp    SMAKYNET
#                          Mike O'Dowd <odowd@ltisun8.epfl.ch>
ntp             123/tcp    Network Time Protocol
ntp             123/udp    Network Time Protocol
#                          Dave Mills <Mills@HUEY.UDEL.EDU>
ansatrader      124/tcp    ANSA REX Trader
ansatrader      124/udp    ANSA REX Trader
#                          Nicola J. Howarth <njh@ansa.co.uk>
locus-map       125/tcp    Locus PC-Interface Net Map Ser
locus-map       125/udp    Locus PC-Interface Net Map Ser
#                          Eric Peterson <lcc.eric@SEAS.UCLA.EDU>
unitary         126/tcp    Unisys Unitary Login
unitary         126/udp    Unisys Unitary Login
#                          <feil@kronos.nisd.cam.unisys.com>
locus-con       127/tcp    Locus PC-Interface Conn Server
locus-con       127/udp    Locus PC-Interface Conn Server
#                          Eric Peterson <lcc.eric@SEAS.UCLA.EDU>
gss-xlicen      128/tcp    GSS X License Verification
gss-xlicen      128/udp    GSS X License Verification
#                          John Light <johnl@gssc.gss.com>
pwdgen          129/tcp    Password Generator Protocol
pwdgen          129/udp    Password Generator Protocol
#               Frank J. Wacho <WANCHO@WSMR-SIMTEL20.ARMY.MIL>
cisco-fna       130/tcp    cisco FNATIVE
cisco-fna       130/udp    cisco FNATIVE
cisco-tna       131/tcp    cisco TNATIVE
cisco-tna       131/udp    cisco TNATIVE
cisco-sys       132/tcp    cisco SYSMAINT
cisco-sys       132/udp    cisco SYSMAINT
statsrv         133/tcp    Statistics Service
statsrv         133/udp    Statistics Service
#                          Dave Mills <Mills@HUEY.UDEL.EDU>
ingres-net      134/tcp    INGRES-NET Service
ingres-net      134/udp    INGRES-NET Service
#                          Mike Berrow <---none--->
loc-srv         135/tcp    Location Service
loc-srv         135/udp    Location Service
#                          Joe Pato <apollo!pato@EDDIE.MIT.EDU>
profile         136/tcp    PROFILE Naming System
profile         136/udp    PROFILE Naming System
#                          Larry Peterson <llp@ARIZONA.EDU>
netbios-ns      137/tcp    NETBIOS Name Service
netbios-ns      137/udp    NETBIOS Name Service
netbios-dgm     138/tcp    NETBIOS Datagram Service
netbios-dgm     138/udp    NETBIOS Datagram Service
netbios-ssn     139/tcp    NETBIOS Session Service
netbios-ssn     139/udp    NETBIOS Session Service
#                          Jon Postel <postel@isi.edu>
emfis-data      140/tcp    EMFIS Data Service
emfis-data      140/udp    EMFIS Data Service
emfis-cntl      141/tcp    EMFIS Control Service
emfis-cntl      141/udp    EMFIS Control Service
#                          Gerd Beling <GBELING@ISI.EDU>
bl-idm          142/tcp    Britton-Lee IDM
bl-idm          142/udp    Britton-Lee IDM
#                          Susie Snitzer <---none--->
imap2           143/tcp    Interim Mail Access Protocol v2
imap2           143/udp    Interim Mail Access Protocol v2
#                          Mark Crispin <MRC@PANDA.COM>
news            144/tcp    NewS
news            144/udp    NewS
#                          James Gosling <JAG@SUN.COM>
```

```
uaac            145/tcp    UAAC Protocol
uaac            145/udp    UAAC Protocol
#               David A. Gomberg <gomberg@GATEWAY.MITRE.ORG>
iso-tp0         146/tcp    ISO-IP0
iso-tp0         146/udp    ISO-IP0
iso-ip          147/tcp    ISO-IP
iso-ip          147/udp    ISO-IP
#                          Marshall Rose <mrose@dbc.mtview.ca.us>
cronus          148/tcp    CRONUS-SUPPORT
cronus          148/udp    CRONUS-SUPPORT
#                          Jeffrey Buffun <jbuffum@APOLLO.COM>
aed-512         149/tcp    AED 512 Emulation Service
aed-512         149/udp    AED 512 Emulation Service
#               Albert G. Broscius <broscius@DSL.CIS.UPENN.EDU>
sql-net         150/tcp    SQL-NET
sql-net         150/udp    SQL-NET
#                          Martin Picard <<---none--->
hems            151/tcp    HEMS
hems            151/udp    HEMS
#                          Christopher Tengi <tengi@Princeton.EDU>
bftp            152/tcp    Background File Transfer Program
bftp            152/udp    Background File Transfer Program
#                          Annette DeSchon <DESCHON@ISI.EDU>
sgmp            153/tcp    SGMP
sgmp            153/udp    SGMP
#                          Marty Schoffstahl <schoff@NISC.NYSER.NET>
netsc-prod      154/tcp    NETSC
netsc-prod      154/udp    NETSC
netsc-dev       155/tcp    NETSC
netsc-dev       155/udp    NETSC
#                          Sergio Heker <heker@JVNCC.CSC.ORG>
sqlsrv          156/tcp    SQL Service
sqlsrv          156/udp    SQL Service
#                          Craig Rogers <Rogers@ISI.EDU>
knet-cmp        157/tcp    KNET/VM Command/Message Protocol
knet-cmp        157/udp    KNET/VM Command/Message Protocol
#                          Gary S. Malkin <GMALKIN@XYLOGICS.COM>
pcmail-srv      158/tcp    PCMail Server
pcmail-srv      158/udp    PCMail Server
#                          Mark L. Lambert <markl@PTT.LCS.MIT.EDU>
nss-routing     159/tcp    NSS-Routing
nss-routing     159/udp    NSS-Routing
#                          Yakov Rekhter <Yakov@IBM.COM>
sgmp-traps      160/tcp    SGMP-TRAPS
sgmp-traps      160/udp    SGMP-TRAPS
#                          Marty Schoffstahl <schoff@NISC.NYSER.NET>
snmp            161/tcp    SNMP
snmp            161/udp    SNMP
snmptrap        162/tcp    SNMPTRAP
snmptrap        162/udp    SNMPTRAP
#                          Marshall Rose <mrose@dbc.mtview.ca.us>
cmip-man        163/tcp    CMIP/TCP Manager
cmip-man        163/udp    CMIP/TCP Manager
cmip-agent      164/tcp    CMIP/TCP Agent
smip-agent      164/udp    CMIP/TCP Agent
#                          Amatzia Ben-Artzi <---none--->
xns-courier     165/tcp    Xerox
xns-courier     165/udp    Xerox
#                          Susie Armstrong <Armstrong.wbst128@XEROX.COM>
s-net           166/tcp    Sirius Systems
s-net           166/udp    Sirius Systems
#                          Brian Lloyd <---none--->
namp            167/tcp    NAMP
namp            167/udp    NAMP
#                          Marty Schoffstahl <schoff@NISC.NYSER.NET>
rsvd            168/tcp    RSVD
rsvd            168/udp    RSVD
#                          Neil Todd <mcvax!ist.co.uk!neil@UUNET.UU.NET>
send            169/tcp    SEND
send            169/udp    SEND
#               William D. Wisner <wisner@HAYES.FAI.ALASKA.EDU>
```

```
print-srv       170/tcp    Network PostScript
print-srv       170/udp    Network PostScript
#                          Brian Reid <reid@DECWRL.DEC.COM>
multiplex       171/tcp    Network Innovations Multiplex
multiplex       171/udp    Network Innovations Multiplex
cl/1            172/tcp    Network Innovations CL/1
cl/1            172/udp    Network Innovations CL/1
#                          Kevin DeVault <<---none--->
xyplex-mux      173/tcp    Xyplex
xyplex-mux      173/udp    Xyplex
#                          Bob Stewart <STEWART@XYPLEX.COM>
mailq           174/tcp    MAILQ
mailq           174/udp    MAILQ
#                          Rayan Zachariassen <rayan@AI.TORONTO.EDU>
vmnet           175/tcp    VMNET
vmnet           175/udp    VMNET
#                          Christopher Tengi <tengi@Princeton.EDU>
genrad-mux      176/tcp    GENRAD-MUX
genrad-mux      176/udp    GENRAD-MUX
#                          Ron Thornton <thornton@qm7501.genrad.com>
xdmcp           177/tcp    X Display Manager Control Protocol
xdmcp           177/udp    X Display Manager Control Protocol
#                          Robert W. Scheifler <RWS@XX.LCS.MIT.EDU>
nextstep        178/tcp    NextStep Window Server
NextStep        178/udp    NextStep Window Server
#                          Leo Hourvitz <leo@NEXT.COM>
bgp             179/tcp    Border Gateway Protocol
bgp             179/udp    Border Gateway Protocol
#                          Kirk Lougheed <LOUGHEED@MATHOM.CISCO.COM>
ris             180/tcp    Intergraph
ris             180/udp    Intergraph
#                          Dave Buehmann <ingr!daveb@UUNET.UU.NET>
unify           181/tcp    Unify
unify           181/udp    Unify
#                          Vinod Singh <--none--->
audit           182/tcp    Unisys Audit SITP
audit           182/udp    Unisys Audit SITP
#                          Gil Greenbaum <gcole@nisd.cam.unisys.com>
ocbinder        183/tcp    OCBinder
ocbinder        183/udp    OCBinder
ocserver        184/tcp    OCServer
ocserver        184/udp    OCServer
#                          Jerrilynn Okamura <--none--->
remote-kis      185/tcp    Remote-KIS
remote-kis      185/udp    Remote-KIS
kis             186/tcp    KIS Protocol
kis             186/udp    KIS Protocol
#                          Ralph Droms <rdroms@NRI.RESTON.VA.US>
aci             187/tcp    Application Communication Interface
aci             187/udp    Application Communication Interface
#                          Rick Carlos <rick.ticipa.csc.ti.com>
mumps           188/tcp    Plus Five's MUMPS
mumps           188/udp    Plus Five's MUMPS
#                          Hokey Stenn <hokey@PLUS5.COM>
qft             189/tcp    Queued File Transport
qft             189/udp    Queued File Transport
#                          Wayne Schroeder <schroeder@SDS.SDSC.EDU>
gacp            190/tcp    Gateway Access Control Protocol
cacp            190/udp    Gateway Access Control Protocol
#                          C. Philip Wood <cpw@LANL.GOV>
prospero        191/tcp    Prospero Directory Service
prospero        191/udp    Prospero Directory Service
#                          B. Clifford Neuman <bcn@isi.edu>
osu-nms         192/tcp    OSU Network Monitoring System
osu-nms         192/udp    OSU Network Monitoring System
#               Doug Karl <KARL-D@OSU-20.IRCC.OHIO-STATE.EDU>
srmp            193/tcp    Spider Remote Monitoring Protocol
srmp            193/udp    Spider Remote Monitoring Protocol
#                          Ted J. Socolofsky <Teds@SPIDER.CO.UK>
irc             194/tcp    Internet Relay Chat Protocol
irc             194/udp    Internet Relay Chat Protocol
```

```
#                          Jarkko Oikarinen <jto@TOLSUN.OULU.FI>
dn6-nlm-aud     195/tcp    DNSIX Network Level Module Audit
dn6-nlm-aud     195/udp    DNSIX Network Level Module Audit
dn6-smm-red     196/tcp    DNSIX Session Mgt Module Audit Redir
dn6-smm-red     196/udp    DNSIX Session Mgt Module Audit Redir
#                          Lawrence Lebahn <DIA3@PAXRV-NES.NAVY.MIL>
dls             197/tcp    Directory Location Service
dls             197/udp    Directory Location Service
dls-mon         198/tcp    Directory Location Service Monitor
dls-mon         198/udp    Directory Location Service Monitor
#                          Scott Bellew <smb@cs.purdue.edu>
smux            199/tcp    SMUX
smux            199/udp    SMUX
#                          Marshall Rose <mrose@dbc.mtview.ca.us>
src             200/tcp    IBM System Resource Controller
src             200/udp    IBM System Resource Controller
#                          Gerald McBrearty <---none--->
at-rtmp         201/tcp    AppleTalk Routing Maintenance
at-rtmp         201/udp    AppleTalk Routing Maintenance
at-nbp          202/tcp    AppleTalk Name Binding
at-nbp          202/udp    AppleTalk Name Binding
at-3            203/tcp    AppleTalk Unused
at-3            203/udp    AppleTalk Unused
at-echo         204/tcp    AppleTalk Echo
at-echo         204/udp    AppleTalk Echo
at-5            205/tcp    AppleTalk Unused
at-5            205/udp    AppleTalk Unused
at-zis          206/tcp    AppleTalk Zone Information
at-zis          206/udp    AppleTalk Zone Information
at-7            207/tcp    AppleTalk Unused
at-7            207/udp    AppleTalk Unused
at-8            208/tcp    AppleTalk Unused
at-8            208/udp    AppleTalk Unused
#                          Rob Chandhok <chandhok@gnome.cs.cmu.edu>
tam             209/tcp    Trivial Authenticated Mail Protocol
tam             209/udp    Trivial Authenticated Mail Protocol
#                          Dan Bernstein <brnstnd@stealth.acf.nyu.edu>
z39.50          210/tcp    ANSI Z39.50
z39.50          210/udp    ANSI Z39.50
#                          Mark Needleman
#                         <mhnur%uccmvsa.bitnet@cornell.cit.cornell.edu>
914c/g          211/tcp    Texas Instruments 914C/G Terminal
914c/g          211/udp    Texas Instruments 914C/G Terminal
#                          Bill Harrell <---none--->
anet            212/tcp    ATEXSSTR
anet            212/udp    ATEXSSTR
#                          Jim Taylor <taylor@heart.epps.kodak.com>
ipx             213/tcp    IPX
ipx             213/udp    IPX
#                          Don Provan <donp@xlnvax.novell.com>
vmpwscs         214/tcp    VM PWSCS
vmpwscs         214/udp    VM PWSCS
#                          Dan Shia <dset!shia@uunet.UU.NET>
softpc          215/tcp    Insignia Solutions
softpc          215/udp    Insignia Solutions
#                          Martyn Thomas <---none--->
atls            216/tcp    Access Technology License Server
atls            216/udp    Access Technology License Server
#                          Larry DeLuca <henrik@EDDIE.MIT.EDU>
dbase           217/tcp    dBASE Unix
dbase           217/udp    dBASE Unix
#                          Don Gibson
#            <sequent!aero!twinsun!ashtate.A-T.COM!dong@uunet.UU.NET>
mpp             218/tcp    Netix Message Posting Protocol
mpp             218/udp    Netix Message Posting Protocol
#                          Shannon Yeh <yeh@netix.com>
uarps           219/tcp    Unisys ARPs
uarps           219/udp    Unisys ARPs
#                          Ashok Marwaha <---none--->
imap3           220/tcp    Interactive Mail Access Protocol v3
imap3           220/udp    Interactive Mail Access Protocol v3
```

```
#                          James Rice <RICE@SUMEX-AIM.STANFORD.EDU>
fln-spx         221/tcp    Berkeley rlogind with SPX auth
fln-spx         221/udp    Berkeley rlogind with SPX auth
rsh-spx         222/tcp    Berkeley rshd with SPX auth
rsh-spx         222/udp    Berkeley rshd with SPX auth
cdc             223/tcp    Certificate Distribution Center
cdc             223/udp    Certificate Distribution Center
#               Kannan Alagappan <kannan@sejour.enet.dec.com>
#               224-241    Reserved
#                          Jon Postel <postel@isi.edu>
#               242/tcp    Unassigned
#               242/udp    Unassigned
sur-meas        243/tcp    Survey Measurement
sur-meas        243/udp    Survey Measurement
#                          Dave Clark <ddc@LCS.MIT.EDU>
#               244/tcp    Unassigned
#               244/udp    Unassigned
link            245/tcp    LINK
link            245/udp    LINK
dsp3270         246/tcp    Display Systems Protocol
dsp3270         246/udp    Display Systems Protocol
#                          Weldon J. Showalter <Gamma@MINTAKA.DCA.MIL>
#               247-255    Reserved
#                          Jon Postel <postel@isi.edu>
#               256-343    Unassigned
pdap            344/tcp    Prospero Data Access Protocol
pdap            344/udp    Prospero Data Access Protocol
#                          B. Clifford Neuman <bcn@isi.edu>
pawserv         345/tcp    Perf Analysis Workbench
pawserv         345/udp    Perf Analysis Workbench
zserv           346/tcp    Zebra server
zserv           346/udp    Zebra server
fatserv         347/tcp    Fatmen Server
fatserv         347/udp    Fatmen Server
csi-sgwp        348/tcp    Cabletron Management Protocol
csi-sgwp        348/udp    Cabletron Management Protocol
#               349-370    Unassigned
clearcase       371/tcp    Clearcase
clearcase       371/udp    Clearcase
#                          Dave LeBlang <leglang@atria.com>
ulistserv       372/tcp    Unix Listserv
ulistserv       372/udp    Unix Listserv
#                          Anastasios Kotsikonas <tasos@cs.bu.edu>
legent-1        373/tcp    Legent Corporation
legent-1        373/udp    Legent Corporation
legent-2        374/tcp    Legent Corporation
legent-2        374/udp    Legent Corporation
#                          Keith Boyce <---none--->
hassle          375/tcp    Hassle
hassle          375/udp    Hassle
#                          Reinhard Doelz <doelz@comp.bioz.unibas.ch>
nip             376/tcp    Amiga Envoy Network Inquiry Proto
nip             376/udp    Amiga Envoy Network Inquiry Proto
#                          Kenneth Dyke <kcd@cbmvax.cbm.commodore.com>
tnETOS          377/tcp    NEC Corporation
tnETOS          377/udp    NEC Corporation
dsETOS          378/tcp    NEC Corporation
dsETOS          378/udp    NEC Corporation
#                          Tomoo Fujita <tf@arc.bs1.fc.nec.co.jp>
is99c           379/tcp    TIA/EIA/IS-99 modem client
is99c           379/udp    TIA/EIA/IS-99 modem client
is99s           380/tcp    TIA/EIA/IS-99 modem server
is99s           380/udp    TIA/EIA/IS-99 modem server
#                          Frank Quick <fquick@qualcomm.com>
hp-collector    381/tcp    hp performance data collector
hp-collector    381/udp    hp performance data collector
hp-managed-node 382/tcp    hp performance data managed node
hp-managed-node 382/udp    hp performance data managed node
hp-alarm-mgr    383/tcp    hp performance data alarm manager
hp-alarm-mgr    383/udp    hp performance data alarm manager
#                          Frank Blakely <frankb@hpptc16.rose.hp.com>
```

```
arns            384/tcp    A Remote Network Server System
arns            384/udp    A Remote Network Server System
#                          David Hornsby <djh@munnari.OZ.AU>
ibm-app         385/tcp    IBM Application
ibm-app         385/tcp    IBM Application
#                          Lisa Tomita <---none--->
asa             386/tcp    ASA Message Router Object Def.
asa             386/udp    ASA Message Router Object Def.
#                          Steve Laitinen <laitinen@brutus.aa.ab.com>
aurp            387/tcp    Appletalk Update-Based Routing Pro.
aurp            387/udp    Appletalk Update-Based Routing Pro.
#                          Chris Ranch <cranch@novell.com>
unidata-ldm     388/tcp    Unidata LDM Version 4
unidata-ldm     388/udp    Unidata LDM Version 4
#                          Glenn Davis <davis@unidata.ucar.edu>
ldap            389/tcp    Lightweight Directory Access Protocol
ldap            389/udp    Lightweight Directory Access Protocol
#                          Tim Howes <Tim.Howes@terminator.cc.umich.edu>
uis             390/tcp    UIS
uis             390/udp    UIS
#                          Ed Barron <---none--->
synotics-relay  391/tcp    SynOptics SNMP Relay Port
synotics-relay  391/udp    SynOptics SNMP Relay Port
synotics-broker 392/tcp    SynOptics Port Broker Port
synotics-broker 392/udp    SynOptics Port Broker Port
#                          Illan Raab <iraab@synoptics.com>
dis             393/tcp    Data Interpretation System
dis             393/udp    Data Interpretation System
#                          Paul Stevens <pstevens@chinacat.Metaphor.COM>
embl-ndt        394/tcp    EMBL Nucleic Data Transfer
embl-ndt        394/udp    EMBL Nucleic Data Transfer
#                          Peter Gad <peter@bmc.uu.se>
netcp           395/tcp    NETscout Control Protocol
netcp           395/udp    NETscout Control Protocol
#                          Anil Singhal <---none--->
netware-ip      396/tcp    Novell Netware over IP
netware-ip      396/udp    Novell Netware over IP
mptn            397/tcp    Multi Protocol Trans. Net.
mptn            397/udp    Multi Protocol Trans. Net.
#                          Soumitra Sarkar <sarkar@vnet.ibm.com>
kryptolan       398/tcp    Kryptolan
kryptolan       398/udp    Kryptolan
#                          Peter de Laval <pdl@sectra.se>
#               399/tcp    Unassigned
#               399/udp    Unassigned
work-sol        400/tcp    Workstation Solutions
work-sol        400/udp    Workstation Solutions
#                          Jim Ward <jimw@worksta.com>
ups             401/tcp    Uninterruptible Power Supply
ups             401/udp    Uninterruptible Power Supply
#                          Guenther Seybold <gs@hrz.th-darmstadt.de>
genie           402/tcp    Genie Protocol
genie           402/udp    Genie Protocol
#                          Mark Hankin <---none--->
decap           403/tcp    decap
decap           403/udp    decap
nced            404/tcp    nced
nced            404/udp    nced
ncld            405/tcp    ncld
ncld            405/udp    ncld
#                          Richard Jones <---none--->
imsp            406/tcp    Interactive Mail Support Protocol
imsp            406/udp    Interactive Mail Support Protocol
#                          John Myers <jgm+@cmu.edu>
timbuktu        407/tcp    Timbuktu
timbuktu        407/udp    Timbuktu
#                          Marc Epard <marc@waygate.farallon.com>
prm-sm          408/tcp    Prospero Resource Manager Sys. Man.
prm-sm          408/udp    Prospero Resource Manager Sys. Man.
prm-nm          409/tcp    Prospero Resource Manager Node Man.
prm-nm          409/udp    Prospero Resource Manager Node Man.
```

```
#                          B. Clifford Neuman <bcn@isi.edu>
decladebug      410/tcp    DECLadebug Remote Debug Protocol
decladebug      410/udp    DECLadebug Remote Debug Protocol
#                          Anthony Berent <berent@rdgeng.enet.dec.com>
rmt             411/tcp    Remote MT Protocol
rmt             411/udp    Remote MT Protocol
#                          Peter Eriksson <pen@lysator.liu.se>
synoptics-trap  412/tcp    Trap Convention Port
synoptics-trap  412/udp    Trap Convention Port
#                          Illan Raab <iraab@synoptics.com>
smsp            413/tcp    SMSP
smsp            413/udp    SMSP
infoseek        414/tcp    InfoSeek
infoseek        414/udp    InfoSeek
#                          Steve Kirsch <stk@frame.com>
bnet            415/tcp    BNet
bnet            415/udp    BNet
#                          Jim Mertz <JMertz+RV09@rvdc.unisys.com>
silverplatter   416/tcp    Silverplatter
silverplatter   416/udp    Silverplatter
#                          Peter Ciuffetti <petec@silverplatter.com>
onmux           417/tcp    Onmux
onmux           417/udp    Onmux
#                          Stephen Hanna <hanna@world.std.com>
hyper-g         418/tcp    Hyper-G
hyper-g         418/udp    Hyper-G
#                          Frank Kappe <fkappe@iicm.tu-graz.ac.at>
ariel1          419/tcp    Ariel
ariel1          419/udp    Ariel
#                          Jonathan Lavigne <BL.JPL@RLG.Stanford.EDU>
smpte           420/tcp    SMPTE
smpte           420/udp    SMPTE
#                          Si Becker <71362.22@CompuServe.COM>
ariel2          421/tcp    Ariel
ariel2          421/udp    Ariel
ariel3          422/tcp    Ariel
ariel3          422/udp    Ariel
#                          Jonathan Lavigne  <BL.JPL@RLG.Stanford.EDU>
opc-job-start   423/tcp    IBM Operations Planning and Control Start
opc-job-start   423/udp    IBM Operations Planning and Control Start
opc-job-track   424/tcp    IBM Operations Planning and Control Track
opc-job-track   424/udp    IBM Operations Planning and Control Track
#                          Conny Larsson  <cocke@VNET.IBM.COM>
icad-el         425/tcp    ICAD
icad-el         425/udp    ICAD
#                          Larry Stone  <lcs@icad.com>
smartsdp        426/tcp    smartsdp
smartsdp        426/udp    smartsdp
#                          Alexander Dupuy <dupuy@smarts.com>
svrloc          427/tcp    Server Location
svrloc          427/udp    Server Location
#                          <veizades@ftp.com>
ocs_cmu         428/tcp    OCS_CMU
ocs_cmu         428/udp    OCS_CMU
ocs_amu         429/tcp    OCS_AMU
ocs_amu         429/udp    OCS_AMU
#                          Florence Wyman <wyman@peabody.plk.af.mil>
utmpsd          430/tcp    UTMPSD
utmpsd          430/udp    UTMPSD
utmpcd          431/tcp    UTMPCD
utmpcd          431/udp    UTMPCD
iasd            432/tcp    IASD
iasd            432/udp    IASD
#                          Nir Baroz <nbaroz@encore.com>
nnsp            433/tcp    NNSP
nnsp            433/udp    NNSP
#                          Rob Robertson <rob@gangrene.berkeley.edu>
mobileip-agent  434/tcp    MobileIP-Agent
mobileip-agent  434/udp    MobileIP-Agent
mobilip-mn      435/tcp    MobilIP-MN
mobilip-mn      435/udp    MobilIP-MN
```

```
#                          Kannan Alagappan <kannan@sejour.lkg.dec.com>
dna-cml         436/tcp    DNA-CML
dna-cml         436/udp    DNA-CML
#                          Dan Flowers <flowers@smaug.lkg.dec.com>
comscm          437/tcp    comscm
comscm          437/udp    comscm
#                          Jim Teague <teague@zso.dec.com>
dsfgw           438/tcp    dsfgw
dsfgw           438/udp    dsfgw
#                          Andy McKeen <mckeen@osf.org>
dasp            439/tcp    dasp      Thomas Obermair
dasp            439/udp    dasp      tommy@inlab.m.eunet.de
#                          Thomas Obermair <tommy@inlab.m.eunet.de>
sgcp            440/tcp    sgcp
sgcp            440/udp    sgcp
#                          Marshall Rose <mrose@dbc.mtview.ca.us>
decvms-sysmgt   441/tcp    decvms-sysmgt
decvms-sysmgt   441/udp    decvms-sysmgt
#                          Lee Barton <barton@star.enet.dec.com>
cvc_hostd       442/tcp    cvc_hostd
cvc_hostd       442/udp    cvc_hostd
#                          Bill Davidson <billd@equalizer.cray.com>
https           443/tcp    https  MCom
https           443/udp    https  MCom
#                          Kipp E.B. Hickman <kipp@mcom.com>
snpp            444/tcp    Simple Network Paging Protocol
snpp            444/udp    Simple Network Paging Protocol
#                          [RFC1568]
microsoft-ds    445/tcp    Microsoft-DS
microsoft-ds    445/udp    Microsoft-DS
#                          Arnold Miller <arnoldm@microsoft.com>
ddm-rdb         446/tcp    DDM-RDB
ddm-rdb         446/udp    DDM-RDB
ddm-dfm         447/tcp    DDM-RFM
ddm-dfm         447/udp    DDM-RFM
ddm-byte        448/tcp    DDM-BYTE
ddm-byte        448/udp    DDM-BYTE
#                          Jan David Fisher <jdfisher@VNET.IBM.COM>
as-servermap    449/tcp    AS Server Mapper
as-servermap    449/udp    AS Server Mapper
#                          Barbara Foss <BGFOSS@rchvmv.vnet.ibm.com>
tserver         450/tcp    TServer
tserver         450/udp    TServer
#                          Harvey S. Schultz <hss@mtgzfs3.mt.att.com>
#               451-511    Unassigned
exec            512/tcp    remote process execution;
#                          authentication performed using
#                          passwords and UNIX loppgin names
biff            512/udp    used by mail system to notify users
#                          of new mail received; currently
#                          receives messages only from
#                          processes on the same machine
login           513/tcp    remote login a la telnet;
#                          automatic authentication performed
#                          based on priviledged port numbers
#                          and distributed data bases which
#                          identify "authentication domains"
who             513/udp    maintains data bases showing who's
#                          logged in to machines on a local
#                          net and the load average of the
#                          machine
cmd             514/tcp    like exec, but automatic
#                          authentication is performed as for
#                          login server
syslog          514/udp
printer         515/tcp    spooler
printer         515/udp    spooler
#               516/tcp    Unassigned
#               516/udp    Unassigned
talk            517/tcp    like tenex link, but across
#                          machine - unfortunately, doesn't
#                          use link protocol (this is actually
#                          just a rendezvous port from which a
#                          tcp connection is established)
talk            517/udp    like tenex link, but across
#                          machine - unfortunately, doesn't
#                          use link protocol (this is actually
#                          just a rendezvous port from which a
                           tcp connection is established)
ntalk           518/tcp
ntalk           518/udp
utime           519/tcp    unixtime
utime           519/udp    unixtime
efs             520/tcp    extended file name server
router          520/udp    local routing process (on site);
#                          uses variant of Xerox NS routing
#                          information protocol
#               521-524    Unassigned
timed           525/tcp    timeserver
timed           525/udp    timeserver
tempo           526/tcp    newdate
tempo           526/udp    newdate
#               527-529    Unassigned
courier         530/tcp    rpc
courier         530/udp    rpc
conference      531/tcp    chat
conference      531/udp    chat
netnews         532/tcp    readnews
netnews         532/udp    readnews
netwall         533/tcp    for emergency broadcasts
netwall         533/udp    for emergency broadcasts
#               534-538    Unassigned
apertus-ldp     539/tcp    Apertus Technologies Load Determination
apertus-ldp     539/udp    Apertus Technologies Load Determination
uucp            540/tcp    uucpd
uucp            540/udp    uucpd
uucp-rlogin     541/tcp    uucp-rlogin  Stuart Lynne
uucp-rlogin     541/udp    uucp-rlogin  sl@wimsey.com
#               542/tcp    Unassigned
#               542/udp    Unassigned
klogin          543/tcp
klogin          543/udp
kshell          544/tcp    krcmd
kshell          544/udp    krcmd
#               545-549    Unassigned
new-rwho        550/tcp    new-who
new-rwho        550/udp    new-who
#               551-555    Unassigned
dsf             555/tcp
dsf             555/udp
remotefs        556/tcp    rfs server
remotefs        556/udp    rfs server
#               557-559    Unassigned
rmonitor        560/tcp    rmonitord
rmonitor        560/udp    rmonitord
monitor         561/tcp
monitor         561/udp
chshell         562/tcp    chcmd
chshell         562/udp    chcmd
#               563/tcp    Unassigned
#               563/udp    Unassigned
9pfs            564/tcp    plan 9 file service
9pfs            564/udp    plan 9 file service
whoami          565/tcp    whoami
whoami          565/udp    whoami
#               566-569    Unassigned
meter           570/tcp    demon
meter           570/udp    demon
meter           571/tcp    udemon
meter           571/udp    udemon
#               572-599    Unassigned
ipcserver       600/tcp    Sun IPC server
ipcserver       600/udp    Sun IPC server
nqs             607/tcp    nqs
nqs             607/udp    nqs
urm             606/tcp    Cray Unified Resource Manager
urm             606/udp    Cray Unified Resource Manager
#                          Bill Schiefelbein <schief@aspen.cray.com>
sift-uft        608/tcp    Sender-Initiated/Unsolicited File Transfer
sift-uft        608/udp    Sender-Initiated/Unsolicited File Transfer
#                          Rick Troth <troth@rice.edu>
npmp-trap       609/tcp    npmp-trap
npmp-trap       609/udp    npmp-trap
npmp-local      610/tcp    npmp-local
npmp-local      610/udp    npmp-local
npmp-gui        611/tcp    npmp-gui
npmp-gui        611/udp    npmp-gui
#                          John Barnes <jbarnes@crl.com>
ginad           634/tcp    ginad
ginad           634/udp    ginad
#                          Mark Crother <mark@eis.calstate.edu>
mdqs            666/tcp
mdqs            666/udp
doom            666/tcp    doom Id Software
doom            666/tcp    doom Id Software
#                          <ddt@idcube.idsoftware.com>
elcsd           704/tcp    errlog copy/server daemon
elcsd           704/udp    errlog copy/server daemon
#
entrustmanager  709/tcp    EntrustManager
entrustmanager  709/udp    EntrustManager
#                          Peter Whittaker <pww@bnr.ca>
netviewdm1      729/tcp    IBM NetView DM/6000 Server/Client
netviewdm1      729/udp    IBM NetView DM/6000 Server/Client
netviewdm2      730/tcp    IBM NetView DM/6000 send/tcp
netviewdm2      730/udp    IBM NetView DM/6000 send/tcp
netviewdm3      731/tcp    IBM NetView DM/6000 receive/tcp
netviewdm3      731/udp    IBM NetView DM/6000 receive/tcp
#                          Philippe Binet  (phbinet@vnet.IBM.COM)
netgw           741/tcp    netGW
netgw           741/udp    netGW
netrcs          742/tcp    Network based Rev. Cont. Sys.
netrcs          742/udp    Network based Rev. Cont. Sys.
#                          Gordon C. Galligher <gorpong@ping.chi.il.us>
flexlm          744/tcp    Flexible License Manager
flexlm          744/udp    Flexible License Manager
#                          Matt Christiano
#                          <globes@matt@oliveb.atc.olivetti.com>
fujitsu-dev     747/tcp    Fujitsu Device Control
fujitsu-dev     747/udp    Fujitsu Device Control
ris-cm          748/tcp    Russell Info Sci Calendar Manager
ris-cm          748/udp    Russell Info Sci Calendar Manager
kerberos-adm    749/tcp    kerberos administration
kerberos-adm    749/udp    kerberos administration
rfile           750/tcp
loadav          750/udp
pump            751/tcp
pump            751/udp
qrh             752/tcp
qrh             752/udp
rrh             753/tcp
rrh             753/udp
tell            754/tcp     send
tell            754/udp     send
nlogin          758/tcp
nlogin          758/udp
con             759/tcp
con             759/udp
ns              760/tcp
ns              760/udp
rxe             761/tcp
rxe             761/udp
quotad          762/tcp
quotad          762/udp
cycleserv       763/tcp
cycleserv       763/udp
omserv          764/tcp
omserv          764/udp
webster         765/tcp
webster         765/udp
phonebook       767/tcp    phone
phonebook       767/udp    phone
vid             769/tcp
vid             769/udp
cadlock         770/tcp
cadlock         770/udp
rtip            771/tcp
rtip            771/udp
cycleserv2      772/tcp
cycleserv2      772/udp
submit          773/tcp
notify          773/udp
rpasswd         774/tcp
acmaint_dbd     774/udp
entomb          775/tcp
acmaint_transd  775/udp
wpages          776/tcp
wpages          776/udp
wpgs            780/tcp
wpgs            780/udp
concert         786/tcp       Concert
concert         786/udp       Concert
#                             Josyula R. Rao <jrrao@watson.ibm.com>
mdbs_daemon     800/tcp
mdbs_daemon     800/udp
device          801/tcp
device          801/udp
xtreelic        996/tcp        Central Point Software
xtreelic        996/udp        Central Point Software
#                              Dale Cabell <dacabell@smtp.xtree.com>
maitrd          997/tcp
maitrd          997/udp
busboy          998/tcp
puparp          998/udp
garcon          999/tcp
applix          999/udp        Applix ac
puprouter       999/tcp
puprouter       999/udp
cadlock         1000/tcp
ock             1000/udp
                1023/tcp       Reserved
                1024/udp       Reserved
#                              IANA <iana@isi.edu>
```

REGISTERED PORT NUMBERS

The Registered Ports are not controlled by the IANA and on most systems can be used by ordinary user processes or programs executed by ordinary users.

Ports are used in the TCP [RFC793] to name the ends of logical connections which carry long term conversations. For the purpose of providing services to unknown callers, a service contact port is defined. This list specifies the port used by the server process as its contact port. While the IANA can not control uses of these ports it does register or list uses of these ports as a convienence to the community.

To the extent possible, these same port assignments are used with the UDP [RFC768].

The Registered Ports are in the range 1024-65535.

Port Assignments:

```
Keyword         Decimal    Description                     References
-------         -------    -----------                     ----------
                1024/tcp   Reserved
                1024/udp   Reserved
#                          IANA <iana@isi.edu>
blackjack       1025/tcp   network blackjack
blackjack       1025/udp   network blackjack
iad1            1030/tcp   BBN IAD
iad1            1030/udp   BBN IAD
iad2            1031/tcp   BBN IAD
iad2            1031/udp   BBN IAD
iad3            1032/tcp   BBN IAD
iad3            1032/udp   BBN IAD
#                          Andy Malis <malis_a@timeplex.com>
instl_boots     1067/tcp   Installation Bootstrap Proto. Serv.
instl_boots     1067/udp   Installation Bootstrap Proto. Serv.
instl_bootc     1068/tcp   Installation Bootstrap Proto. Cli.
instl_bootc     1068/udp   Installation Bootstrap Proto. Cli.
#                          David Arko <<darko@hpfcrn.fc.hp.com>
socks           1080/tcp   Socks
socks           1080/udp   Socks
#                          Ying-Da Lee <ylee@syl.dl.nec.com
ansoft-lm-1     1083/tcp   Anasoft License Manager
ansoft-lm-1     1083/udp   Anasoft License Manager
ansoft-lm-2     1084/tcp   Anasoft License Manager
ansoft-lm-2     1084/udp   Anasoft License Manager
nfa             1155/tcp   Network File Access
nfa             1155/udp   Network File Access
#                          James Powell <james@mailhost.unidata.com>
nerv            1222/tcp   SNI R&D network
nerv            1222/udp   SNI R&D network
#                          Martin Freiss <freiss.pad@sni.de>
hermes          1248/tcp
hermes          1248/udp
alta-ana-lm     1346/tcp   Alta Analytics License Manager
alta-ana-lm     1346/udp   Alta Analytics License Manager
bbn-mmc         1347/tcp   multi media conferencing
bbn-mmc         1347/udp   multi media conferencing
bbn-mmx         1348/tcp   multi media conferencing
bbn-mmx         1348/udp   multi media conferencing
sbook           1349/tcp   Registration Network Protocol
sbook           1349/udp   Registration Network Protocol
editbench       1350/tcp   Registration Network Protocol
editbench       1350/udp   Registration Network Protocol
#               Simson L. Garfinkel <simsong@next.cambridge.ma.us>
equationbuilder 1351/tcp   Digital Tool Works (MIT)
equationbuilder 1351/udp   Digital Tool Works (MIT)
#                          Terrence J. Talbot <lexcube!tjt@bu.edu>
lotusnote       1352/tcp   Lotus Note
lotusnote       1352/udp   Lotus Note
#               Greg Pflaum <iris.com!Greg_Pflaum@uunet.uu.net>
relief          1353/tcp   Relief Consulting
relief          1353/udp   Relief Consulting
#                          John Feiler <relief!jjfeiler@uu2.psi.com>
rightbrain      1354/tcp   RightBrain Software
rightbrain      1354/udp   RightBrain Software
#                          Glenn Reid <glann@rightbrain.com>
intuitive edge  1355/tcp   Intuitive Edge
intuitive edge  1355/udp   Intuitive Edge
#                          Montgomery Zukowski
#                          <monty@nextnorth.acs.ohio-state.edu>
cuillamartin    1356/tcp   CuillaMartin Company
cuillamartin    1356/udp   CuillaMartin Company
pegboard        1357/tcp   Electronic PegBoard
pegboard        1357/udp   Electronic PegBoard
#                          Chris Cuilla
#                          <balr!vpnet!cuilla!chris@clout.chi.il.us>
```

```
connlcli        1358/tcp   CONNLCLI
connlcli        1358/udp   CONNLCLI
ftsrv           1359/tcp   FTSRV
ftsrv           1359/udp   FTSRV
#                          Ines Homem de Melo <sidinf@brfapesp.bitnet>
mimer           1360/tcp   MIMER
mimer           1360/udp   MIMER
#                          Per Schroeder  <Per.Schroder@mimer.se>
linx            1361/tcp   LinX
linx            1361/udp   LinX
#                          Steffen Schilke <---none--->
timeflies       1362/tcp   TimeFlies
timeflies       1362/udp   TimeFlies
#                          Doug Kent <mouthers@slugg@nwnexus.wa.com>
ndm-requester   1363/tcp   Network DataMover Requester
ndm-requester   1363/udp   Network DataMover Requester
ndm-server      1364/tcp   Network DataMover Server
ndm-server      1364/udp   Network DataMover Server
#                          Toshio Watanabe
#                          <watanabe@godzilla.rsc.spdd.ricoh.co.j>
adapt-sna       1365/tcp   Network Software Associates
adapt-sna       1365/udp   Network Software Associates
#                          Jeffery Chiao <714-768-401>
netware-csp     1366/tcp   Novell NetWare Comm Service Platform
netware-csp     1366/udp   Novell NetWare Comm Service Platform
#                          Laurie Lindsey <llindsey@novell.com>
dcs             1367/tcp   DCS
dcs             1367/udp   DCS
#                          Stefan Siebert <ssiebert@dcs.de>
screencast      1368/tcp   ScreenCast
screencast      1368/udp   ScreenCast
#                          Bill Tschumy <other!bill@uunet.UU.NET>
gv-us           1369/tcp   GlobalView to Unix Shell
gv-us           1369/udp   GlobalView to Unix Shell
us-gv           1370/tcp   Unix Shell to GlobalView
us-gv           1370/udp   Unix Shell to GlobalView
#               Makoto Mita <mita@ssdev.ksp.fujixerox.co.jp>
fc-cli          1371/tcp   Fujitsu Config Protocol
fc-cli          1371/udp   Fujitsu Config Protocol
fc-ser          1372/tcp   Fujitsu Config Protocol
fc-ser          1372/udp   Fujitsu Config Protocol
#               Ryuichi Horie <horie@spad.sysrap.cs.fujitsu.co.jp>
chromagrafx     1373/tcp   Chromagrafx
chromagrafx     1373/udp   Chromagrafx
#                          Mike Barthelemy <msb@chromagrafx.com>
molly           1374/tcp   EPI Software Systems
molly           1374/udp   EPI Software Systems
#                          Jim Vlcek <vlcek@epimbe.com>
bytex           1375/tcp   Bytex
bytex           1375/udp   Bytex
#               Mary Ann Burt <bytex!ws054!maryann@uunet.UU.NET>
ibm-pps         1376/tcp   IBM Person to Person Software
ibm-pps         1376/udp   IBM Person to Person Software
#                          Simon Phipps <sphipps@vnet.ibm.com>
cichlid         1377/tcp   Cichlid License Manager
cichlid         1377/udp   Cichlid License Manager
#                          Andy Burgess <aab@cichlid.com>
elan            1378/tcp   Elan License Manager
elan            1378/udp   Elan License Manager
#                          Ken Greer <kg@elan.com>
dbreporter      1379/tcp   Integrity Solutions
dbreporter      1379/udp   Integrity Solutions
#                          Tim Dawson <tdawson%mspboss@uunet.UU.NET>
telesis-licman  1380/tcp   Telesis Network License Manager
telesis-licman  1380/udp   Telesis Network License Manager
#                          Karl Schendel, Jr. <wiz@telesis.com>
apple-licman    1381/tcp   Apple Network License Manager
apple-licman    1381/udp   Apple Network License Manager
#                          Earl Wallace <earlw@apple.com>
udt_os          1382/tcp
udt_os          1382/udp
```

```
gwha            1383/tcp   GW Hannaway Network License Manager
gwha            1383/udp   GW Hannaway Network License Manager
#                          J. Gabriel Foster <fop@gwha.com>
os-licman       1384/tcp   Objective Solutions License Manager
os-licman       1384/udp   Objective Solutions License Manager
#               Donald Cornwell <don.cornwell@objective.com>
atex_elmd       1385/tcp   Atex Publishing License Manager
atex_elmd       1385/udp   Atex Publishing License Manager
#                          Brett Sorenson <bcs@atex.com>
checksum        1386/tcp   CheckSum License Manager
checksum        1386/udp   CheckSum License Manager
#                          Andreas Glocker <glocker@sirius.com>
cadsi-lm        1387/tcp   Computer Aided Design Software Inc LM
cadsi-lm        1387/udp   Computer Aided Design Software Inc LM
#                          Sulistio Muljadi
objective-dbc   1388/tcp   Objective Solutions DataBase Cache
objective-dbc   1388/udp   Objective Solutions DataBase Cache
#                          Donald Cornwell
iclpv-dm        1389/tcp   Document Manager
iclpv-dm        1389/udp   Document Manager
iclpv-sc        1390/tcp   Storage Controller
iclpv-sc        1390/udp   Storage Controller
iclpv-sas       1391/tcp   Storage Access Server
iclpv-sas       1391/udp   Storage Access Server
iclpv-pm        1392/tcp   Print Manager
iclpv-pm        1392/udp   Print Manager
iclpv-nls       1393/tcp   Network Log Server
iclpv-nls       1393/udp   Network Log Server
iclpv-nlc       1394/tcp   Network Log Client
iclpv-nlc       1394/udp   Network Log Client
iclpv-wsm       1395/tcp   PC Workstation Manager software
iclpv-wsm       1395/udp   PC Workstation Manager software
#               A.P. Hobson <A.P.Hobson@bra0112.wins.icl.co.uk>
dvl-activemail  1396/tcp   DVL Active Mail
dvl-activemail  1396/udp   DVL Active Mail
audio-activmail 1397/tcp   Audio Active Mail
audio-activmail 1397/udp   Audio Active Mail
video-activmail 1398/tcp   Video Active Mail
video-activmail 1398/udp   Video Active Mail
#                          Ehud Shapiro <udi@wisdon.weizmann.ac.il>
cadkey-licman   1399/tcp   Cadkey License Manager
cadkey-licman   1399/udp   Cadkey License Manager
cadkey-tablet   1400/tcp   Cadkey Tablet Daemon
cadkey-tablet   1400/udp   Cadkey Tablet Daemon
#                          Joe McCollough <joe@cadkey.com>
goldleaf-licman 1401/tcp   Goldleaf License Manager
goldleaf-licman 1401/udp   Goldleaf License Manager
#                          John Fox <---none--->
prm-sm-np       1402/tcp   Prospero Resource Manager
prm-sm-np       1402/udp   Prospero Resource Manager
prm-nm-np       1403/tcp   Prospero Resource Manager
prm-nm-np       1403/udp   Prospero Resource Manager
#                          B. Clifford Neuman <bcn@isi.edu>
igi-lm          1404/tcp   Infinite Graphics License Manager
igi-lm          1404/udp   Infinite Graphics License Manager
ibm-res         1405/tcp   IBM Remote Execution Starter
ibm-res         1405/udp   IBM Remote Execution Starter
netlabs-lm      1406/tcp   NetLabs License Manager
netlabs-lm      1406/udp   NetLabs License Manager
dbsa-lm         1407/tcp   DBSA License Manager
dbsa-lm         1407/udp   DBSA License Manager
#                          Scott Shattuck <ss@dbsa.com>
sophia-lm       1408/tcp   Sophia License Manager
sophia-lm       1408/udp   Sophia License Manager
#                          Eric Brown <sst!emerald!eric@uunet.UU.net>
here-lm         1409/tcp   Here License Manager
here-lm         1409/udp   Here License Manager
#                          David Ison  <here@dialup.oar.net>
hiq             1410/tcp   HiQ License Manager
hiq             1410/udp   HiQ License Manager
#                          Rick Pugh <rick@bilmillennium.com>
```

```
af              1411/tcp   AudioFile
af              1411/udp   AudioFile
#                          Jim Gettys <jg@crl.dec.com>
innosys         1412/tcp   InnoSys
innosys         1412/udp   InnoSys
innosys-acl     1413/tcp   Innosys-ACL
innosys-acl     1413/udp   Innosys-ACL
#                          Eric Welch <--none--->
ibm-mqseries    1414/tcp   IBM MQSeries
ibm-mqseries    1414/udp   IBM MQSeries
#                          Roger Meli <rmmeli%winvmd@vnet.ibm.com>
dbstar          1415/tcp   DBStar
dbstar          1415/udp   DBStar
#                          Jeffrey Millman <jcm@dbstar.com>
novell-lu6.2    1416/tcp   Novell LU6.2
novell-lu6.2    1416/udp   Novell LU6.2
#                          Peter Liu <--none--->
timbuktu-srv1   1417/tcp   Timbuktu Service 1 Port
timbuktu-srv1   1417/tcp   Timbuktu Service 1 Port
timbuktu-srv2   1418/tcp   Timbuktu Service 2 Port
timbuktu-srv2   1418/udp   Timbuktu Service 2 Port
timbuktu-srv3   1419/tcp   Timbuktu Service 3 Port
timbuktu-srv3   1419/udp   Timbuktu Service 3 Port
timbuktu-srv4   1420/tcp   Timbuktu Service 4 Port
timbuktu-srv4   1420/udp   Timbuktu Service 4 Port
#                          Marc Epard <marc@waygate.farallon.com>
gandalf-lm      1421/tcp   Gandalf License Manager
gandalf-lm      1421/udp   Gandalf License Manager
#                          gilmer@gandalf.ca
autodesk-lm     1422/tcp   Autodesk License Manager
autodesk-lm     1422/udp   Autodesk License Manager
#                          David Ko <dko@autodesk.com>
essbase         1423/tcp   Essbase Arbor Software
essbase         1423/udp   Essbase Arbor Software
hybrid          1424/tcp   Hybrid Encryption Protocol
hybrid          1424/udp   Hybrid Encryption Protocol
#                          Howard Hart <hch@hybrid.com>
zion-lm         1425/tcp   Zion Software License Manager
zion-lm         1425/udp   Zion Software License Manager
#                          David Ferrero <david@zion.com>
sas-1           1426/tcp   Satellite-data Acquisition System 1
sas-1           1426/udp   Satellite-data Acquisition System 1
#                          Bill Taylor <sais@ssec.wisc.edu>
mloadd          1427/tcp   mloadd monitoring tool
mloadd          1427/udp   mloadd monitoring tool
#                          Bob Braden <braden@isi.edu>
informatik-lm   1428/tcp   Informatik License Manager
informatik-lm   1428/udp   Informatik License Manager
#                          Harald Schlangmann
#                          <schlangm@informatik.uni-muenchen.de>
nms             1429/tcp   Hypercom NMS
nms             1429/udp   Hypercom NMS
tpdu            1430/tcp   Hypercom TPDU
tpdu            1430/udp   Hypercom TPDU
#                          Noor Chowdhury <noor@hypercom.com>
rgtp            1431/tcp   Reverse Gosip Transport
rgtp            1431/udp   Reverse Gosip Transport
#                          <iwj10@cl.cam-orl.co.uk>
blueberry-lm    1432/tcp   Blueberry Software License Manager
blueberry-lm    1432/udp   Blueberry Software License Manager
#                          Steve Beigel <ublueb!steve@uunet.uu.net>
ms-sql-s        1433/tcp   Microsoft-SQL-Server
ms-sql-s        1433/udp   Microsoft-SQL-Server
ms-sql-m        1434/tcp   Microsoft-SQL-Monitor
ms-sql-m        1434/udp   Microsoft-SQL-Monitor
#                          Peter Hussey <peterhus@microsoft.com>
ibm-cics        1435/tcp   IBM CISC
ibm-cics        1435/udp   IBM CISC
#                          Geoff Meacock <gbibmswl@ibmmail.COM>
sas-2           1436/tcp   Satellite-data Acquisition System 2
sas-2           1436/udp   Satellite-data Acquisition System 2
```

```
#                          Bill Taylor <sais@ssec.wisc.edu>
tabula          1437/tcp   Tabula
tabula          1437/udp   Tabula
#                          Marcelo Einhorn
#                          <KGUNE%HUJIVM1.bitnet@taunivm.tau.ac.il>
eicon-server    1438/tcp   Eicon Security Agent/Server
eicon-server    1438/udp   Eicon Security Agent/Server
eicon-x25       1439/tcp   Eicon X25/SNA Gateway
eicon-x25       1439/udp   Eicon X25/SNA Gateway
eicon-slp       1440/tcp   Eicon Service Location Protocol
eicon-slp       1440/udp   Eicon Service Location Protocol
#                          Pat Calhoun <CALHOUN@admin.eicon.qc.ca>
cadis-1         1441/tcp   Cadis License Management
cadis-1         1441/udp   Cadis License Management
cadis-2         1442/tcp   Cadis License Management
cadis-2         1442/udp   Cadis License Management
#                          Todd Wichers <twichers@csn.org>
ies-lm          1443/tcp   Integrated Engineering Software
ies-lm          1443/udp   Integrated Engineering Software
#                          David Tong <David_Tong@integrated.mb.ca>
marcam-lm       1444/tcp   Marcam  License Management
marcam-lm       1444/udp   Marcam  License Management
#                          Therese Hunt <hunt@marcam.com>
proxima-lm      1445/tcp   Proxima License Manager
proxima-lm      1445/udp   Proxima License Manager
ora-lm          1446/tcp   Optical Research Associates License Manager
ora-lm          1446/udp   Optical Research Associates License Manager
apri-lm         1447/tcp   Applied Parallel Research LM
apri-lm         1447/udp   Applied Parallel Research LM
#                          Jim Dillon <jed@apri.com>
oc-lm           1448/tcp   OpenConnect License Manager
oc-lm           1448/udp   OpenConnect License Manager
#                          Sue Barnhill <snb@oc.com>
peport          1449/tcp   PEport
peport          1449/udp   PEport
#                          Qentin Neill <quentin@ColumbiaSC.NCR.COM>
dwf             1450/tcp   Tandem Distributed Workbench Facility
dwf             1450/udp   Tandem Distributed Workbench Facility
#                          Mike Bert <BERG_MIKE@tandem.com>
infoman         1451/tcp   IBM Information Management
infoman         1451/udp   IBM Information Management
#                          Karen Burns <---none--->
gtegsc-lm       1452/tcp   GTE Government Systems License Man
gtegsc-lm       1452/udp   GTE Government Systems License Man
#               Mike Gregory <Gregory_Mike@msmail.iipo.gtegsc.com>
genie-lm        1453/tcp   Genie License Manager
genie-lm        1453/udp   Genie License Manager
#                          Paul Applegate <p.applegate2@genie.geis.com>
interhdl_elmd   1454/tcp   interHDL License Manager
interhdl_elmd   1454/tcp   interHDL License Manager
#                          Eli Sternheim eli@interhdl.com
esl-lm          1455/tcp   ESL License Manager
esl-lm          1455/udp   ESL License Manager
#                          Abel Chou <abel@willy.esl.com>
dca             1456/tcp   DCA
dca             1456/udp   DCA
#                          Jeff Garbers <jgarbers@netcom.com>
valisys-lm      1457/tcp    Valisys License Manager
valisys-lm      1457/udp    Valisys License Manager
#               Leslie Lincoln <leslie_lincoln@valisys.com>
nrcabq-lm       1458/tcp    Nichols Research Corp.
nrcabq-lm       1458/udp    Nichols Research Corp.
#                           Howard Cole <hcole@tumbleweed.nrcabq.com>
proshare1       1459/tcp    Proshare Notebook Application
proshare1       1459/udp    Proshare Notebook Application
proshare2       1460/tcp    Proshare Notebook Application
proshare2       1460/udp    Proshare Notebook Application
#                           Robin Kar <Robin_Kar@ccm.hf.intel.com>
ibm_wrless_lan  1461/tcp    IBM Wireless LAN
ibm_wrless_lan  1461/udp    IBM Wireless LAN
#                           <flanne@vnet.IBM.COM>
```

```
world-lm        1462/tcp    World License Manager
world-lm        1462/udp    World License Manager
#                           Michael S Amirault <ambi@world.std.com>
nucleus         1463/tcp    Nucleus
nucleus         1463/udp    Nucleus
#                           Venky Nagar <venky@fafner.Stanford.EDU>
msl_lmd         1464/tcp    MSL License Manager
msl_lmd         1464/udp    MSL License Manager
#                           Matt Timmermans
pipes           1465/tcp    Pipes Platform
pipes           1465/udp    Pipes Platform  mfarlin@peerlogic.com
#                           Mark Farlin <mfarlin@peerlogic.com>
oceansoft-lm    1466/tcp    Ocean Software License Manager
oceansoft-lm    1466/udp    Ocean Software License Manager
#                           Randy Leonard <randy@oceansoft.com>
csdmbase        1467/tcp    CSDMBASE
csdmbase        1467/udp    CSDMBASE
csdm            1468/tcp    CSDM
csdm            1468/udp    CSDM
#               Robert Stabl <stabl@informatik.uni-muenchen.de>
aal-lm          1469/tcp    Active Analysis Limited License Manager
aal-lm          1469/udp    Active Analysis Limited License Manager
#                           David Snocken  +44 (71)437-7009
uaiact          1470/tcp    Universal Analytics
uaiact          1470/udp    Universal Analytics
#                           Mark R. Ludwig <Mark-Ludwig@uai.com>
csdmbase        1471/tcp    csdmbase
csdmbase        1471/udp    csdmbase
csdm            1472/tcp    csdm
csdm            1472/udp    csdm
#               Robert Stabl <stabl@informatik.uni-muenchen.de>
openmath        1473/tcp    OpenMath
openmath        1473/udp    OpenMath
#                           Garth Mayville <mayville@maplesoft.on.ca>
telefinder      1474/tcp    Telefinder
telefinder      1474/udp    Telefinder
#                           Jim White <Jim_White@spiderisland.com>
taligent-lm     1475/tcp    Taligent License Manager
taligent-lm     1475/udp    Taligent License Manager
#               Mark Sapsford <Mark_Sapsford@@taligent.com>
clvm-cfg        1476/tcp    clvm-cfg
clvm-cfg        1476/udp    clvm-cfg
#                           Eric Soderberg <seric@cup.hp.com>
ms-sna-server   1477/tcp    ms-sna-server
ms-sna-server   1477/udp    ms-sna-server
ms-sna-base     1478/tcp    ms-sna-base
ms-sna-base     1478/udp    ms-sna-base
#                           Gordon Mangione <gordm@microsoft.com>
dberegister     1479/tcp    dberegister
dberegister     1479/udp    dberegister
#                           Brian Griswold <brian@dancingbear.com>
pacerforum      1480/tcp    PacerForum
pacerforum      1480/udp    PacerForum
#                           Peter Caswell <pfc@pacvax.pacersoft.com>
airs            1481/tcp    AIRS
airs            1481/udp    AIRS
#                           Bruce Wilson, 905-771-6161
miteksys-lm     1482/tcp    Miteksys License Manager
miteksys-lm     1482/udp    Miteksys License Manager
#                           Shane McRoberts <mcroberts@miteksys.com>
afs             1483/tcp    AFS License Manager
afs             1483/udp    AFS License Manager
#                           Michael R. Pizolato <michael@afs.com>
confluent       1484/tcp    Confluent License Manager
confluent       1484/udp    Confluent License Manager
#                           James Greenfiel <jim@pa.confluent.com>
lansource       1485/tcp    LANSource
lansource       1485/udp    LANSource
#                           Doug Scott <lansourc@hookup.net>
nms_topo_serv   1486/tcp    nms_topo_serv
nms_topo_serv   1486/udp    nms_topo_serv
```

```
#                           Sylvia Siu <Sylvia_Siu@Novell.CO>
localinfosrvr   1487/tcp    LocalInfoSrvr
localinfosrvr   1487/udp    LocalInfoSrvr
#               Brian Matthews <brian_matthews@ibist.ibis.com>
docstor         1488/tcp    DocStor
docstor         1488/udp    DocStor
#                           Brian Spears <bspears@salix.com>
dmdocbroker     1489/tcp    dmdocbroker
dmdocbroker     1489/udp    dmdocbroker
#                           Razmik Abnous <abnous@documentum.com>
insitu-conf     1490/tcp    insitu-conf
insitu-conf     1490/udp    insitu-conf
#                           Paul Blacknell <paul@insitu.com>
anynetgateway   1491/tcp    anynetgateway
anynetgateway   1491/udp    anynetgateway
#                           Dan Poirier <poirier@VNET.IBM.COM>
stone-design-1  1492/tcp    stone-design-1
stone-design-1  1492/udp    stone-design-1
#                           Andrew Stone <andrew@stone.com>
netmap_lm       1493/tcp    netmap_lm
netmap_lm       1493/udp    netmap_lm
#                           Phillip Magson <philm@extro.ucc.su.OZ.AU>
ica             1494/tcp    ica
ica             1494/udp    ica
#                           John Richardson, Citrix Systems
cvc             1495/tcp    cvc
cvc             1495/udp    cvc
#                           Bill Davidson <billd@equalizer.cray.com>
liberty-lm      1496/tcp    liberty-lm
liberty-lm      1496/udp    liberty-lm
#                           Jim Rogers <trane!jimbo@pacbell.com>
rfx-lm          1497/tcp    rfx-lm
rfx-lm          1497/udp    rfx-lm
#                           Bill Bishop <bil@rfx.rfx.com>
watcom-sql      1498/tcp    Watcom-SQL
watcom-sql      1498/udp    Watcom-SQL
#                           Rog Skubowius <rwskubow@ccnga.uwaterloo.ca>
fhc             1499/tcp    Federico Heinz Consultora
fhc             1499/udp    Federico Heinz Consultora
#                           Federico Heinz <federico@heinz.com>
vlsi-lm         1500/tcp    VLSI License Manager
vlsi-lm         1500/udp    VLSI License Manager
#                           Shue-Lin Kuo <shuelin@mdk.sanjose.vlsi.com>
sas-3           1501/tcp    Satellite-data Acquisition System 3
sas-3           1501/udp    Satellite-data Acquisition System 3
#                           Bill Taylor <sais@ssec.wisc.edu>
shivadiscovery  1502/tcp    Shiva
shivadiscovery  1502/udp    Shiva
#                           Jonathan Wenocur <jhw@Shiva.COM>
imtc-mcs        1503/tcp    Databeam
imtc-mcs        1503/udp    Databeam
#                           Jim Johnstone <jjohnstone@databeam.com>
evb-elm         1504/tcp    EVB Software Engineering License Manager
evb-elm         1504/udp    EVB Software Engineering License Manager
#                           B.G. Mahesh < mahesh@sett.com>
funkproxy       1505/tcp    Funk Software, Inc.
funkproxy       1505/udp    Funk Software, Inc.
#                           Robert D. Vincent <bert@willowpond.com>
#               1506-1523   Unassigned
ingreslock      1524/tcp    ingres
ingreslock      1524/udp    ingres
orasrv          1525/tcp    oracle
orasrv          1525/udp    oracle
prospero-np     1525/tcp    Prospero Directory Service non-priv
prospero-np     1525/udp    Prospero Directory Service non-priv
pdap-np         1526/tcp    Prospero Data Access Prot non-priv
pdap-np         1526/udp    Prospero Data Access Prot non-priv
#                           B. Clifford Neuman <bcn@isi.edu>
tlisrv          1527/tcp    oracle
tlisrv          1527/udp    oracle
coauthor        1529/tcp    oracle
coauthor        1529/udp    oracle
issd            1600/tcp
issd            1600/udp
nkd             1650/tcp
nkd             1650/udp
proshareaudio   1651/tcp   proshare conf audio
proshareaudio   1651/udp   proshare conf audio
prosharevideo   1652/tcp   proshare conf video
prosharevideo   1652/udp   proshare conf video
prosharedata    1653/tcp   proshare conf data
prosharedata    1653/udp   proshare conf data
prosharerequest 1654/tcp   proshare conf request
prosharerequest 1654/udp   proshare conf request
prosharenotify  1655/tcp   proshare conf notify
prosharenotify  1655/udp   proshare conf notify
#                          <gunner@ibeam.intel.com>
netview-aix-1   1661/tcp   netview-aix-1
netview-aix-1   1661/udp   netview-aix-1
netview-aix-2   1662/tcp   netview-aix-2
netview-aix-2   1662/udp   netview-aix-2
netview-aix-3   1663/tcp   netview-aix-3
netview-aix-3   1663/udp   netview-aix-3
netview-aix-4   1664/tcp   netview-aix-4
netview-aix-4   1664/udp   netview-aix-4
netview-aix-5   1665/tcp   netview-aix-5
netview-aix-5   1665/udp   netview-aix-5
netview-aix-6   1666/tcp   netview-aix-6
netview-aix-6   1666/udp   netview-aix-6
#               Martha Crisson <CRISSON@ralvm12.vnet.ibm.com>
licensedaemon   1986/tcp   cisco license management
licensedaemon   1986/udp   cisco license management
tr-rsrb-p1      1987/tcp   cisco RSRB Priority 1 port
tr-rsrb-p1      1987/udp   cisco RSRB Priority 1 port
tr-rsrb-p2      1988/tcp   cisco RSRB Priority 2 port
tr-rsrb-p2      1988/udp   cisco RSRB Priority 2 port
tr-rsrb-p3      1989/tcp   cisco RSRB Priority 3 port
tr-rsrb-p3      1989/udp   cisco RSRB Priority 3 port
#PROBLEMS!===================================================
mshnet          1989/tcp   MHSnet system
mshnet          1989/udp   MHSnet system
#               Bob Kummerfeld <bob@sarad.cs.su.oz.au>
#PROBLEMS!===================================================
stun-p1         1990/tcp   cisco STUN Priority 1 port
stun-p1         1990/udp   cisco STUN Priority 1 port
stun-p2         1991/tcp   cisco STUN Priority 2 port
stun-p2         1991/udp   cisco STUN Priority 2 port
stun-p3         1992/tcp   cisco STUN Priority 3 port
stun-p3         1992/udp   cisco STUN Priority 3 port
#PROBLEMS!===================================================
ipsendmsg       1992/tcp   IPsendmsg
ipsendmsg       1992/udp   IPsendmsg
#               Bob Kummerfeld <bob@sarad.cs.su.oz.au>
#PROBLEMS!===================================================
snmp-tcp-port   1993/tcp   cisco SNMP TCP port
snmp-tcp-port   1993/udp   cisco SNMP TCP port
stun-port       1994/tcp   cisco serial tunnel port
stun-port       1994/udp   cisco serial tunnel port
perf-port       1995/tcp   cisco perf port
perf-port       1995/udp   cisco perf port
tr-rsrb-port    1996/tcp   cisco Remote SRB port
tr-rsrb-port    1996/udp   cisco Remote SRB port
gdp-port        1997/tcp   cisco Gateway Discovery Protocol
gdp-port        1997/udp   cisco Gateway Discovery Protocol
x25-svc-port    1998/tcp   cisco X.25 service (XOT)
x25-svc-port    1998/udp   cisco X.25 service (XOT)
tcp-id-port     1999/tcp   cisco identification port
tcp-id-port     1999/udp   cisco identification port
callbook        2000/tcp
callbook        2000/udp
dc              2001/tcp
wizard          2001/udp    curry
```

```
globe           2002/tcp
globe           2002/udp
mailbox         2004/tcp
emce            2004/udp    CCWS mm conf
berknet         2005/tcp
oracle          2005/udp
invokator       2006/tcp
raid-cc         2006/udp    raid
dectalk         2007/tcp
raid-am         2007/udp
conf            2008/tcp
terminaldb      2008/udp
news            2009/tcp
whosockami      2009/udp
search          2010/tcp
pipe_server     2010/udp
raid-cc         2011/tcp    raid
servserv        2011/udp
ttyinfo         2012/tcp
raid-ac         2012/udp
raid-am         2013/tcp
raid-cd         2013/udp
troff           2014/tcp
raid-sf         2014/udp
cypress         2015/tcp
raid-cs         2015/udp
bootserver      2016/tcp
bootserver      2016/udp
cypress-stat    2017/tcp
bootclient      2017/udp
terminaldb      2018/tcp
rellpack        2018/udp
whosockami      2019/tcp
about           2019/udp
xinupageserver  2020/tcp
xinupageserver  2020/udp
servexec        2021/tcp
xinuexpansion1  2021/udp
down            2022/tcp
xinuexpansion2  2022/udp
xinuexpansion3  2023/tcp
xinuexpansion3  2023/udp
xinuexpansion4  2024/tcp
xinuexpansion4  2024/udp
ellpack         2025/tcp
xribs           2025/udp
scrabble        2026/tcp
scrabble        2026/udp
shadowserver    2027/tcp
shadowserver    2027/udp
submitserver    2028/tcp
submitserver    2028/udp
device2         2030/tcp
device2         2030/udp
blackboard      2032/tcp
blackboard      2032/udp
glogger         2033/tcp
glogger         2033/udp
scoremgr        2034/tcp
scoremgr        2034/udp
imsldoc         2035/tcp
imsldoc         2035/udp
objectmanager   2038/tcp
objectmanager   2038/udp
lam             2040/tcp
lam             2040/udp
interbase       2041/tcp
interbase       2041/udp
isis            2042/tcp
isis            2042/udp
isis-bcast      2043/tcp
```

```
isis-bcast      2043/udp
rimsl           2044/tcp
rimsl           2044/udp
cdfunc          2045/tcp
cdfunc          2045/udp
sdfunc          2046/tcp
sdfunc          2046/udp
dls             2047/tcp
dls             2047/udp
dls-monitor     2048/tcp
dls-monitor     2048/udp
shilp           2049/tcp
shilp           2049/udp
dlsrpn          2065/tcp   Data Link Switch Read Port Number
dlsrpn          2065/udp   Data Link Switch Read Port Number
dlswpn          2067/tcp   Data Link Switch Write Port Number
dlswpn          2067/udp   Data Link Switch Write Port Number
ats             2201/tcp   Advanced Training System Program
ats             2201/udp   Advanced Training System Program
rtsserv         2500/tcp   Resource Tracking system server
rtsserv         2500/udp   Resource Tracking system server
rtsclient       2501/tcp   Resource Tracking system client
rtsclient       2501/udp   Resource Tracking system client
#                          Aubrey Turner
#               <S95525ta%etsuacad.bitnet@ETSUADMN.ETSU.EDU>
hp-3000-telnet  2564/tcp   HP 3000 NS/VT block mode telnet
www-dev         2784/tcp   world wide web - development
www-dev         2784/udp   world wide web - development
NSWS            3049/tcp
NSWS            3049/udp
ccmail          3264/tcp   cc:mail/lotus
ccmail          3264/udp   cc:mail/lotus
dec-notes       3333/tcp   DEC Notes
dec-notes       3333/udp   DEC Notes
#                          Kim Moraros <moraros@via.enet.dec.com>
mapper-nodemgr    3984/tcp     MAPPER network node manager
mapper-nodemgr    3984/udp     MAPPER network node manager
mapper-mapethd    3985/tcp     MAPPER TCP/IP server
mapper-mapethd    3985/udp     MAPPER TCP/IP server
mapper-ws_ethd    3986/tcp     MAPPER workstation server
mapper-ws_ethd    3986/udp     MAPPER workstation server
#                 John C. Horton <jch@unirsvl.rsvl.unisys.com>
bmap            3421/tcp   Bull Apprise portmapper
bmap            3421/udp   Bull Apprise portmapper
#                          Jeremy Gilbert <J.Gilbert@ma30.bull.com>
udt_os          3900/tcp   Unidata UDT OS
udt_os          3900/udp   Unidata UDT OS
#                          James Powell <james@mailhost.unidata.com>
nuts_dem        4132/tcp   NUTS Daemon
nuts_dem        4132/udp   NUTS Daemon
nuts_bootp      4133/tcp   NUTS Bootp Server
nuts_bootp      4133/udp   NUTS Bootp Server
#                          Martin Freiss <freiss.pad@sni.>
unicall         4343/tcp   UNICALL
unicall         4343/udp   UNICALL
#                          James Powell <james@enghp.unidata.comp>
krb524          4444/tcp   KRB524
krb524          4444/udp   KRB524
#                          B. Clifford Neuman <bcn@isi.edu>
rfa             4672/tcp   remote file access server
rfa             4672/udp   remote file access server
commplex-main   5000/tcp
commplex-main   5000/udp
commplex-link   5001/tcp
commplex-link   5001/udp
rfe             5002/tcp   radio free ethernet
rfe             5002/udp   radio free ethernet
telelpathstart  5010/tcp   TelepathStart
telelpathstart  5010/udp   TelepathStart
telelpathattack 5011/tcp   TelepathAttack
telelpathattack 5011/udp   TelepathAttack
```

```
#               Helmuth Breitenfellner <hbreitenf@vnet.imb.com>
mmcc            5050/tcp   multimedia conference control tool
mmcc            5050/udp   multimedia conference control tool
rmonitor_secure 5145/tcp
rmonitor_secure 5145/udp
aol             5190/tcp   America-Online
aol             5190/udp   America-Online
#                          Marty Lyons <marty@aol.com>
padl2sim        5236/tcp
padl2sim        5236/udp
hacl-hb         5300/tcp        # HA cluster heartbeat
hacl-hb         5300/udp        # HA cluster heartbeat
hacl-gs         5301/tcp        # HA cluster general services
hacl-gs         5301/udp        # HA cluster general services
hacl-cfg        5302/tcp        # HA cluster configuration
hacl-cfg        5302/udp        # HA cluster configuration
hacl-probe      5303/tcp        # HA cluster probing
hacl-probe      5303/udp        # HA cluster probing
hacl-local      5304/tcp
hacl-local      5304/udp
hacl-test       5305/tcp
hacl-test       5305/udp
#                               Eric Soderberg <seric@hposl102.cup.hp>
x11             6000-6063/tcp   X Window System
x11             6000-6063/udp   X Window System
#               Stephen Gildea <gildea@expo.lcs.mit.edu>
sub-process     6111/tcp   HP SoftBench Sub-Process Control
sub-process     6111/udp   HP SoftBench Sub-Process Control
meta-corp       6141/tcp   Meta Corporation License Manager
meta-corp       6141/udp   Meta Corporation License Manager
#                          Osamu Masuda <--none--->
aspentec-lm     6142/tcp   Aspen Technology License Manager
aspentec-lm     6142/udp   Aspen Technology License Manager
#                          Kevin Massey <massey@aspentec.com>
watershed-lm    6143/tcp   Watershed License Manager
watershed-lm    6143/udp   Watershed License Manager
#                          David Ferrero <david@zion.com>
statsci1-lm     6144/tcp   StatSci License Manager - 1
statsci1-lm     6144/udp   StatSci License Manager - 1
statsci2-lm     6145/tcp   StatSci License Manager - 2
statsci2-lm     6145/udp   StatSci License Manager - 2
#                          Scott Blachowicz <scott@statsci.com>
lonewolf-lm     6146/tcp   Lone Wolf Systems License Manager
lonewolf-lm     6146/udp   Lone Wolf Systems License Manager
#                          Dan Klein <dvk@lonewolf.com>
montage-lm      6147/tcp   Montage License Manager
montage-lm      6147/udp   Montage License Manager
#                          Michael Ubell <michael@montage.com>
xdsxdm          6558/udp
xdsxdm          6558/tcp
afs3-fileserver 7000/tcp   file server itself
afs3-fileserver 7000/udp   file server itself
afs3-callback   7001/tcp   callbacks to cache managers
afs3-callback   7001/udp   callbacks to cache managers
afs3-prserver   7002/tcp   users & groups database
afs3-prserver   7002/udp   users & groups database
afs3-vlserver   7003/tcp   volume location database
afs3-vlserver   7003/udp   volume location database
afs3-kaserver   7004/tcp   AFS/Kerberos authentication service
afs3-kaserver   7004/udp   AFS/Kerberos authentication service
afs3-volser     7005/tcp   volume managment server
afs3-volser     7005/udp   volume managment server
afs3-errors     7006/tcp   error interpretation service
afs3-errors     7006/udp   error interpretation service
afs3-bos        7007/tcp   basic overseer process
afs3-bos        7007/udp   basic overseer process
afs3-update     7008/tcp   server-to-server updater
afs3-update     7008/udp   server-to-server updater
afs3-rmtsys     7009/tcp   remote cache manager service
afs3-rmtsys     7009/udp   remote cache manager service
ups-onlinet     7010/tcp   onlinet uninterruptable power supplies
```

```
ups-onlinet     7010/udp   onlinet uninterruptable power supplies
#                          Brian Hammill <hamill@dolphin.exide.com>
font-service    7100/tcp   X Font Service
font-service    7100/udp   X Font Service
#                          Stephen Gildea <gildea@expo.lcs.mit.edu>
fodms           7200/tcp   FODMS FLIP
fodms           7200/udp   FODMS FLIP
#           David Anthony <anthony@power.amasd.anatcp.rockwell.com>
man             9535/tcp
man             9535/udp
isode-dua       17007/tcp
isode-dua       17007/udp
```

REFERENCES

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, USC/Information Sciences Institute, August 1980.

[RFC793] Postel, J., ed., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, USC/Information Sciences Institute, September 1981.

[]

URL = ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers

RFC 959 Index

RFC 959

```
Network Working Group                                          J. Postel
Request for Comments: 959                                    J. Reynolds
                                                                     ISI
Obsoletes RFC: 765 (IEN 149)                                October 1985
```

FILE TRANSFER PROTOCOL (FTP)

Status of this Memo

This memo is the official specification of the File Transfer Protocol (FTP). Distribution of this memo is unlimited.

The following new optional commands are included in this edition of the specification:

CDUP (Change to Parent Directory), SMNT (Structure Mount), STOU (Store Unique), RMD (Remove Directory), MKD (Make Directory), PWD (Print Directory), and SYST (System).

Note that this specification is compatible with the previous edition.

1. INTRODUCTION

The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs.

The attempt in this specification is to satisfy the diverse needs of users of maxi-hosts, mini-hosts, personal workstations, and TACs, with a simple, and easily implemented protocol design.

This paper assumes knowledge of the Transmission Control Protocol (TCP) [2] and the Telnet Protocol [3]. These documents are contained in the ARPA-Internet protocol handbook [1].

2. OVERVIEW

In this section, the history, the terminology, and the FTP model are discussed. The terms defined in this section are only those that have special significance in FTP. Some of the terminology is very specific to the FTP model; some readers may wish to turn to the section on the FTP model while reviewing the terminology.

2.1. HISTORY

FTP has had a long evolution over the years. Appendix III is a chronological compilation of Request for Comments documents relating to FTP. These include the first proposed file transfer mechanisms in 1971 that were developed for implementation on hosts at M.I.T. (RFC 114), plus comments and discussion in RFC 141.

RFC 172 provided a user-level oriented protocol for file transfer between host computers (including terminal IMPs). A revision of this as RFC 265, restated FTP for additional review, while RFC 281 suggested further changes. The use of a "Set Data Type" transaction was proposed in RFC 294 in January 1982.

RFC 354 obsoleted RFCs 264 and 265. The File Transfer Protocol was now defined as a protocol for file transfer between HOSTs on the ARPANET, with the primary function of FTP defined as transfering files efficiently and reliably among hosts and allowing the convenient use of remote file storage capabilities. RFC 385 further commented on errors, emphasis points, and additions to the protocol, while RFC 414 provided a status report on the working server and user FTPs. RFC 430, issued in 1973, (among other RFCs too numerous to mention) presented further comments on FTP. Finally, an "official" FTP document was published as RFC 454.

By July 1973, considerable changes from the last versions of FTP were made, but the general structure remained the same. RFC 542 was published as a new "official" specification to reflect these changes. However, many implementations based on the older specification were not updated.

In 1974, RFCs 607 and 614 continued comments on FTP. RFC 624 proposed further design changes and minor modifications. In 1975, RFC 686 entitled, "Leaving Well Enough Alone", discussed the differences between all of the early and later versions of FTP. RFC 691 presented a minor revision of RFC 686, regarding the subject of print files.

Motivated by the transition from the NCP to the TCP as the underlying protocol, a phoenix was born out of all of the above efforts in RFC 765 as the specification of FTP for use on TCP.

This current edition of the FTP specification is intended to correct some minor documentation errors, to improve the explanation of some protocol features, and to add some new optional commands.

In particular, the following new optional commands are included in this edition of the specification:

CDUP - Change to Parent Directory

SMNT - Structure Mount

STOU - Store Unique

RMD - Remove Directory

MKD - Make Directory

PWD - Print Directory

SYST - System

This specification is compatible with the previous edition. A program implemented in conformance to the previous specification should automatically be in conformance to this specification.

2.2. TERMINOLOGY

ASCII

The ASCII character set is as defined in the ARPA-Internet Protocol Handbook. In FTP, ASCII characters are defined to be the lower half of an eight-bit code set (i.e., the most significant bit is zero).

access controls

Access controls define users' access privileges to the use of a system, and to the files in that system. Access controls are necessary to prevent unauthorized or accidental use of files. It is the prerogative of a server-FTP process to invoke access controls.

byte size

There are two byte sizes of interest in FTP: the logical byte size of the file, and the transfer byte size used for the transmission of the data. The transfer byte size is always 8 bits. The transfer byte size is not necessarily the byte size in which data is to be stored in a system, nor the logical byte size for interpretation of the structure of the data.

control connection

The communication path between the USER-PI and SERVER-PI for the exchange of commands and replies. This connection follows the Telnet Protocol.

data connection

A full duplex connection over which data is transferred, in a specified mode and type. The data transferred may be a part of a file, an entire file or a number of files. The path may be between a server-DTP and a user-DTP, or between two server-DTPs.

data port

The passive data transfer process "listens" on the data port for a connection from the active transfer process in order to open the data connection.

DTP

The data transfer process establishes and manages the data connection. The DTP can be passive or active.

End-of-Line

The end-of-line sequence defines the separation of printing lines. The sequence is Carriage Return, followed by Line Feed.

EOF

The end-of-file condition that defines the end of a file being transferred.

EOR

The end-of-record condition that defines the end of a record being transferred.

error recovery

A procedure that allows a user to recover from certain errors such as failure of either host system or transfer process. In FTP, error recovery may involve restarting a file transfer at a given checkpoint.

FTP commands

A set of commands that comprise the control information flowing from the user-FTP to the server-FTP process.

file

An ordered set of computer data (including programs), of arbitrary length, uniquely identified by a pathname.

mode

The mode in which data is to be transferred via the data connection. The mode defines the data format during transfer including EOR and EOF. The transfer modes defined in FTP are described in the Section on Transmission Modes.

NVT

The Network Virtual Terminal as defined in the Telnet Protocol.

NVFS

The Network Virtual File System. A concept which defines a standard network file system with standard commands and pathname conventions.

page

A file may be structured as a set of independent parts called pages. FTP supports the transmission of discontinuous files as independent indexed pages.

pathname

Pathname is defined to be the character string which must be input to a file system by a user in order to identify a file. Pathname normally contains device and/or directory names, and file name specification. FTP does not yet specify a standard pathname convention. Each user must follow the file naming conventions of the file systems involved in the transfer.

PI

The protocol interpreter. The user and server sides of the protocol have distinct roles implemented in a user-PI and a server-PI.

record

A sequential file may be structured as a number of contiguous parts called records. Record structures are supported by FTP but a file need not have record structure.

reply

A reply is an acknowledgment (positive or negative) sent from server to user via the control connection in response to FTP commands. The general form of a reply is a completion code (including error codes) followed by a text string. The codes are for use by programs and the text is usually intended for human users.

server-DTP

The data transfer process, in its normal "active" state, establishes the data connection with the "listening" data port. It sets up parameters for transfer and storage, and transfers data on command from its PI. The DTP can be placed in a "passive" state to listen for, rather than initiate a connection on the data port.

server-FTP process

A process or set of processes which perform the function of file transfer in cooperation with a user-FTP process and, possibly, another server. The functions consist of a protocol interpreter (PI) and a data transfer process (DTP).

server-PI

The server protocol interpreter "listens" on Port L for a connection from a user-PI and establishes a control communication connection. It receives standard FTP commands from the user-PI, sends replies, and governs the server-DTP.

type

The data representation type used for data transfer and storage. Type implies certain transformations between the time of data storage and data transfer. The representation types defined in FTP are described in the Section on Establishing Data Connections.

user

A person or a process on behalf of a person wishing to obtain file transfer service. The human user may interact directly with a server-FTP process, but use of a user-FTP process is preferred since the protocol design is weighted towards automata.

user-DTP

The data transfer process "listens" on the data port for a connection from a server-FTP process. If two servers are transferring data between them, the user-DTP is inactive.

user-FTP process

A set of functions including a protocol interpreter, a data transfer process and a user interface which together perform the function of file transfer in cooperation with one or more server-FTP processes. The user interface allows a local language to be used in the command-reply dialogue with the user.

user-PI

The user protocol interpreter initiates the control connection from its port U to the server-FTP process, initiates FTP commands, and governs the user-DTP if that process is part of the file transfer.

2.3. THE FTP MODEL

With the above definitions in mind, the following model (shown in Figure 1) may be diagrammed for an FTP service.

```
                                      -------------
                                      |/---------\|
                                      ||   User  ||    --------
                                      ||Interface|<--->| User |
                                      |\----^----/|    --------
            ----------                |     |     |
            |/------\|  FTP Commands  |/----V----\|
            ||Server|<---------------->|   User  ||
            ||  PI  ||   FTP Replies  ||    PI   ||
            |\--^---/|                |\----^----/|
            |   |    |                |     |     |
--------    |/--V---\|      Data      |/----V----\|    --------
| File |<--->|Server|<---------------->|  User   |<--->| File |
|System|    || DTP  ||   Connection   ||   DTP   ||    |System|
--------    |\------/|                |\---------/|    --------
            ----------                -------------

            Server-FTP                   USER-FTP
```

NOTES: 1. The data connection may be used in either direction.
       2. The data connection need not exist all of the time.

Figure 1 Model for FTP Use

In the model described in Figure 1, the user-protocol interpreter initiates the control connection. The control connection follows the Telnet protocol. At the initiation of the user, standard FTP commands are generated by the user-PI and transmitted to the server process via the control connection. (The user may establish a direct control connection to the server-FTP, from a TAC terminal for example, and generate standard FTP commands independently, bypassing the user-FTP process.) Standard replies are sent from the server-PI to the user-PI over the control connection in response to the commands.

The FTP commands specify the parameters for the data connection (data port, transfer mode, representation type, and structure) and the nature of file system operation (store, retrieve, append, delete, etc.). The user-DTP or its designate should "listen" on the specified data port, and the server initiate the data connection and data transfer in accordance with the specified parameters. It should be noted that the data port need not be in the same host that initiates the FTP commands via the control connection, but the user or the user-FTP process must ensure a "listen" on the specified data port. It ought to also be noted that the data connection may be used for simultaneous sending and receiving.

In another situation a user might wish to transfer files between two hosts, neither of which is a local host. The user sets up control connections to the two servers and then arranges for a data connection between them. In this manner, control information is passed to the user-PI but data is transferred between the server data transfer processes. Following is a model of this server-server interaction.

```
        Control     ------------   Control
        ---------->| User-FTP |<-----------
        |          | User-PI  |           |
        |          |   "C"    |           |
        V          ------------           V
--------------                        --------------
| Server-FTP |   Data Connection      | Server-FTP |
|    "A"     |<---------------------->|    "B"     |
-------------- Port (A)      Port (B) --------------
```

Figure 2

The protocol requires that the control connections be open while data transfer is in progress. It is the responsibility of the user to request the closing of the control connections when finished using the FTP service, while it is the server who takes the action. The server may abort data transfer if the control connections are closed without command.

The Relationship between FTP and Telnet:

The FTP uses the Telnet protocol on the control connection. This can be achieved in two ways: first, the user-PI or the server-PI may implement the rules of the Telnet Protocol directly in their own procedures; or, second, the user-PI or the server-PI may make use of the existing Telnet module in the system.

Ease of implementaion, sharing code, and modular programming argue for the second approach. Efficiency and independence argue for the first approach. In practice, FTP relies on very little of the Telnet Protocol, so the first approach does not necessarily involve a large amount of code.

3. DATA TRANSFER FUNCTIONS

Files are transferred only via the data connection. The control connection is used for the transfer of commands, which describe the functions to be performed, and the replies to these commands (see the Section on FTP Replies). Several commands are concerned with the transfer of data between hosts. These data transfer commands include the MODE command which specify how the bits of the data are to be transmitted, and the STRUcture and TYPE commands, which are used to define the way in which the data are to be represented. The transmission and representation are basically independent but the "Stream" transmission mode is dependent on the file structure attribute and if "Compressed" transmission mode is used, the nature of the filler byte depends on the representation type.

3.1. DATA REPRESENTATION AND STORAGE

Data is transferred from a storage device in the sending host to a storage device in the receiving host. Often it is necessary to perform certain transformations on the data because data storage representations in the two systems are different. For example, NVT-ASCII has different data storage representations in different systems. DEC TOPS-20s's generally store NVT-ASCII as five 7-bit ASCII characters, left-justified in a 36-bit word. IBM Mainframe's store NVT-ASCII as 8-bit EBCDIC codes. Multics stores NVT-ASCII as four 9-bit characters in a 36-bit word. It is desirable to convert characters into the standard NVT-ASCII representation when transmitting text between dissimilar systems. The sending and receiving sites would have to perform the necessary transformations between the standard representation and their internal representations.

A different problem in representation arises when transmitting binary data (not character codes) between host systems with different word lengths. It is not always clear how the sender should send data, and the receiver store it. For example, when transmitting 32-bit bytes from a 32-bit word-length system to a 36-bit word-length system, it may be desirable (for reasons of efficiency and usefulness) to store the 32-bit bytes right-justified in a 36-bit word in the latter system. In any case, the user should have the option of specifying data representation and transformation functions. It should be noted that FTP provides for very limited data type representations. Transformations desired beyond this limited capability should be performed by the user directly.

3.1.1. DATA TYPES

Data representations are handled in FTP by a user specifying a representation type. This type may implicitly (as in ASCII or EBCDIC) or explicitly (as in Local byte) define a byte size for interpretation which is referred to as the "logical byte size." Note that this has nothing to do with the byte size used for transmission over the data connection, called the "transfer byte size", and the two should not be confused. For example, NVT-ASCII has a logical byte size of 8 bits. If the type is Local byte, then the TYPE command has an obligatory second parameter specifying the logical byte size. The transfer byte size is always 8 bits.

3.1.1.1. ASCII TYPE

This is the default type and must be accepted by all FTP implementations. It is intended primarily for the transfer of text files, except when both hosts would find the EBCDIC type more convenient.

The sender converts the data from an internal character representation to the standard 8-bit NVT-ASCII representation (see the Telnet specification). The receiver will convert the data from the standard form to his own internal form.

In accordance with the NVT standard, the <CRLF> sequence should be used where necessary to denote the end of a line of text. (See the discussion of file structure at the end of the Section on Data Representation and Storage.)

Using the standard NVT-ASCII representation means that data must be interpreted as 8-bit bytes.

The Format parameter for ASCII and EBCDIC types is discussed below.

3.1.1.2. EBCDIC TYPE

This type is intended for efficient transfer between hosts which use EBCDIC for their internal character representation.

For transmission, the data are represented as 8-bit EBCDIC characters. The character code is the only difference between the functional specifications of EBCDIC and ASCII types.

End-of-line (as opposed to end-of-record--see the discussion of structure) will probably be rarely used with EBCDIC type for purposes of denoting structure, but where it is necessary the <NL> character should be used.

3.1.1.3. IMAGE TYPE

The data are sent as contiguous bits which, for transfer, are packed into the 8-bit transfer bytes. The receiving site must store the data as contiguous bits. The structure of the storage system might necessitate the padding of the file (or of each record, for a record-structured file) to some convenient boundary (byte, word or block). This padding, which must be all zeros, may occur only at the end of the file (or at the end of each record) and there must be a way of identifying the padding bits so that they may be stripped off if the file is retrieved. The padding transformation should be well publicized to enable a user to process a file at the storage site.

Image type is intended for the efficient storage and retrieval of files and for the transfer of binary data. It is recommended that this type be accepted by all FTP implementations.

3.1.1.4. LOCAL TYPE

The data is transferred in logical bytes of the size specified by the obligatory second parameter, Byte size. The value of Byte size must be a decimal integer; there is no default value. The logical byte size is not necessarily the same as the transfer byte size. If there is a difference in byte sizes, then the logical bytes should be packed contiguously, disregarding transfer byte boundaries and with any necessary padding at the end.

When the data reaches the receiving host, it will be transformed in a manner dependent on the logical byte size and the particular host. This transformation must be invertible (i.e., an identical file can be retrieved if the same parameters are used) and should be well publicized by the FTP implementors.

For example, a user sending 36-bit floating-point numbers to a host with a 32-bit word could send that data as Local byte with a logical byte size of 36. The receiving host would then be expected to store the logical bytes so that they could be easily manipulated; in this example putting the

36-bit logical bytes into 64-bit double words should suffice.

In another example, a pair of hosts with a 36-bit word size may send data to one another in words by using TYPE L 36. The data would be sent in the 8-bit transmission bytes packed so that 9 transmission bytes carried two host words.

3.1.1.5. FORMAT CONTROL

The types ASCII and EBCDIC also take a second (optional) parameter; this is to indicate what kind of vertical format control, if any, is associated with a file. The following data representation types are defined in FTP:

A character file may be transferred to a host for one of three purposes: for printing, for storage and later retrieval, or for processing. If a file is sent for printing, the receiving host must know how the vertical format control is represented. In the second case, it must be possible to store a file at a host and then retrieve it later in exactly the same form. Finally, it should be possible to move a file from one host to another and process the file at the second host without undue trouble. A single ASCII or EBCDIC format does not satisfy all these conditions. Therefore, these types have a second parameter specifying one of the following three formats:

3.1.1.5.1. NON PRINT

This is the default format to be used if the second (format) parameter is omitted. Non-print format must be accepted by all FTP implementations.

The file need contain no vertical format information. If it is passed to a printer process, this process may assume standard values for spacing and margins.

Normally, this format will be used with files destined for processing or just storage.

3.1.1.5.2. TELNET FORMAT CONTROLS

The file contains ASCII/EBCDIC vertical format controls (i.e., <CR>, <LF>, <NL>, <VT>, <FF>) which the printer process will interpret appropriately. <CRLF>, in exactly this sequence, also denotes end-of-line.

3.1.1.5.2. CARRIAGE CONTROL (ASA)

The file contains ASA (FORTRAN) vertical format control characters. (See RFC 740 Appendix C; and Communications of the ACM, Vol. 7, No. 10, p. 606, October 1964.) In a line or a record formatted according to the ASA Standard, the first character is not to be printed. Instead, it should be used to determine the vertical movement of the paper which should take place before the rest of the record is printed.

The ASA Standard specifies the following control characters:

Character	Vertical Spacing
blank	Move paper up one line
0	Move paper up two lines
1	Move paper to top of next page
+	No movement, i.e., overprint

Clearly there must be some way for a printer process to distinguish the end of the structural entity. If a file has record structure (see below) this is no problem; records will be explicitly marked during transfer and storage. If the file has no record structure, the <CRLF> end-of-line sequence is used to separate printing lines, but these format effectors are overridden by the ASA controls.

3.1.2. DATA STRUCTURES

In addition to different representation types, FTP allows the structure of a file to be specified. Three file structures are defined in FTP:

file-structure, where there is no internal structure and the file is considered to be a continuous sequence of data bytes,

record-structure, where the file is made up of sequential records,

and page-structure, where the file is made up of independent indexed pages.

File-structure is the default to be assumed if the STRUcture command has not been used but both file and record structures must be accepted for "text" files (i.e., files with TYPE ASCII or EBCDIC) by all FTP implementations. The structure of a file will affect both the transfer mode of a file (see the Section on Transmission Modes) and the interpretation and storage of the file.

The "natural" structure of a file will depend on which host stores the file. A source-code file will usually be stored on an IBM Mainframe in fixed length records but on a DEC TOPS-20 as a stream of characters partitioned into lines, for example by <CRLF>. If the transfer of files between such disparate sites is to be useful, there must be some way for one site to recognize the other's assumptions about the file.

With some sites being naturally file-oriented and others naturally record-oriented there may be problems if a file with one structure is sent to a host oriented to the other. If a text file is sent with record-structure to a host which is file oriented, then that host should apply an internal transformation to the file based on the record structure. Obviously, this transformation should be useful, but it must also be invertible so that an identical file may be retrieved using record structure.

In the case of a file being sent with file-structure to a record-oriented host, there exists the question of what criteria the host should use to divide the file into records which can be processed locally. If this division is necessary, the FTP implementation should use the end-of-line sequence,

<CRLF> for ASCII, or <NL> for EBCDIC text files, as the delimiter. If an FTP implementation adopts this technique, it must be prepared to reverse the transformation if the file is retrieved with file-structure.

3.1.2.1. FILE STRUCTURE

File structure is the default to be assumed if the STRUcture command has not been used.

In file-structure there is no internal structure and the file is considered to be a continuous sequence of data bytes.

3.1.2.2. RECORD STRUCTURE

Record structures must be accepted for "text" files (i.e., files with TYPE ASCII or EBCDIC) by all FTP implementations.

In record-structure the file is made up of sequential records.

3.1.2.3. PAGE STRUCTURE

To transmit files that are discontinuous, FTP defines a page structure. Files of this type are sometimes known as "random access files" or even as "holey files". In these files there is sometimes other information associated with the file as a whole (e.g., a file descriptor), or with a section of the file (e.g., page access controls), or both. In FTP, the sections of the file are called pages.

To provide for various page sizes and associated information, each page is sent with a page header. The page header has the following defined fields:

Header Length

The number of logical bytes in the page header including this byte. The minimum header length is 4.

Page Index

The logical page number of this section of the file. This is not the transmission sequence number of this page, but the index used to identify this page of the file.

Data Length

The number of logical bytes in the page data. The minimum data length is 0.

Page Type

The type of page this is. The following page types are defined:

0 = Last Page

This is used to indicate the end of a paged structured transmission. The header length must be 4, and the data length must be 0.

1 = Simple Page

This is the normal type for simple paged files with no page level associated control information. The header length must be 4.

2 = Descriptor Page

This type is used to transmit the descriptive information for the file as a whole.

3 = Access Controlled Page

This type includes an additional header field for paged files with page level access control information. The header length must be 5.

Optional Fields

Further header fields may be used to supply per page control information, for example, per page access control.

All fields are one logical byte in length. The logical byte size is specified by the TYPE command. See Appendix I for further details and a specific case at the page structure.

A note of caution about parameters: a file must be stored and retrieved with the same parameters if the retrieved version is to be identical to the version originally transmitted. Conversely, FTP implementations must return a file identical to the original if the parameters used to store and retrieve a file are the same.

3.2. ESTABLISHING DATA CONNECTIONS

The mechanics of transferring data consists of setting up the data connection to the appropriate ports and choosing the parameters for transfer. Both the user and the server-DTPs have a default data port. The user-process default data port is the same as the control connection port (i.e., U). The server-process default data port is the port adjacent to the control connection port (i.e., L-1).

The transfer byte size is 8-bit bytes. This byte size is relevant only for the actual transfer of the data; it has no bearing on representation of the data within a host's file system.

The passive data transfer process (this may be a user-DTP or a second server-DTP) shall "listen" on the data port prior to sending a transfer request command. The FTP request command determines the direction of the data transfer. The server, upon receiving the transfer request, will initiate the data connection to the port. When the connection is established, the data transfer begins between DTP's, and the server-PI sends a confirming reply to the user-PI.

Every FTP implementation must support the use of the default data ports, and only the USER-PI can initiate a change to non-default ports.

It is possible for the user to specify an alternate data port by use of the PORT command. The user may want a file dumped on a TAC line printer or retrieved from a third party host. In the latter case, the user-PI sets up control connections with both server-PI's. One server is then told (by an FTP command) to "listen" for a connection which the other will initiate. The user-PI sends one server-PI a PORT command indicating the data port of the other. Finally, both are sent the appropriate transfer commands. The exact sequence of commands and replies sent between the user-controller and the servers is defined in the Section on FTP Replies.

In general, it is the server's responsibility to maintain the data connection--to initiate it and to close it. The exception to this is when the user-DTP is sending the data in a transfer mode that requires the connection to be closed to indicate EOF. The server MUST close the data connection under the following conditions:

1. The server has completed sending data in a transfer mode that requires a close to indicate EOF.

2. The server receives an ABORT command from the user.

3. The port specification is changed by a command from the user.

4. The control connection is closed legally or otherwise.

5. An irrecoverable error condition occurs.

Otherwise the close is a server option, the exercise of which the server must indicate to the user-process by either a 250 or 226 reply only.

3.3. DATA CONNECTION MANAGEMENT

Default Data Connection Ports: All FTP implementations must support use of the default data connection ports, and only the User-PI may initiate the use of non-default ports.

Negotiating Non-Default Data Ports: The User-PI may specify a non-default user side data port with the PORT command. The User-PI may request the server side to identify a non-default server side data port with the PASV command. Since a connection is defined by the pair of addresses, either of these actions is enough to get a different data connection, still it is permitted to do both commands to use new ports on both ends of the data connection.

Reuse of the Data Connection: When using the stream mode of data transfer the end of the file must be indicated by closing the connection. This causes a problem if multiple files are to be transfered in the session, due to need for TCP to hold the connection record for a time out period to guarantee the reliable communication. Thus the connection can not be reopened at once.

There are two solutions to this problem. The first is to negotiate a non-default port. The second is to use another transfer mode.

A comment on transfer modes. The stream transfer mode is

inherently unreliable, since one can not determine if the connection closed prematurely or not. The other transfer modes (Block, Compressed) do not close the connection to indicate the end of file. They have enough FTP encoding that the data connection can be parsed to determine the end of the file. Thus using these modes one can leave the data connection open for multiple file transfers.

3.4. TRANSMISSION MODES

The next consideration in transferring data is choosing the appropriate transmission mode. There are three modes: one which formats the data and allows for restart procedures; one which also compresses the data for efficient transfer; and one which passes the data with little or no processing. In this last case the mode interacts with the structure attribute to determine the type of processing. In the compressed mode, the representation type determines the filler byte.

All data transfers must be completed with an end-of-file (EOF) which may be explicitly stated or implied by the closing of the data connection. For files with record structure, all the end-of-record markers (EOR) are explicit, including the final one. For files transmitted in page structure a "last-page" page type is used.

NOTE: In the rest of this section, byte means "transfer byte" except where explicitly stated otherwise.

For the purpose of standardized transfer, the sending host will translate its internal end of line or end of record denotation into the representation prescribed by the transfer mode and file structure, and the receiving host will perform the inverse translation to its internal denotation. An IBM Mainframe record count field may not be recognized at another host, so the end-of-record information may be transferred as a two byte control code in Stream mode or as a flagged bit in a Block or Compressed mode descriptor. End-of-line in an ASCII or EBCDIC file with no record structure should be indicated by <CRLF> or <NL>, respectively. Since these transformations imply extra work for some systems, identical systems transferring non-record structured text files might wish to use a binary representation and stream mode for the transfer.

The following transmission modes are defined in FTP:

3.4.1. STREAM MODE

The data is transmitted as a stream of bytes. There is no restriction on the representation type used; record structures are allowed.

In a record structured file EOR and EOF will each be indicated by a two-byte control code. The first byte of the control code will be all ones, the escape character. The second byte will have the low order bit on and zeros elsewhere for EOR and the second low order bit on for EOF; that is, the byte will have value 1 for EOR and value 2 for EOF. EOR and EOF may be indicated together on the last byte transmitted by turning both low order bits on (i.e., the value 3). If a byte of all ones was intended to be sent as data, it should be repeated in the second byte of the control code.

If the structure is a file structure, the EOF is indicated by the sending host closing the data connection and all bytes are data bytes.

3.4.2. BLOCK MODE

The file is transmitted as a series of data blocks preceded by one or more header bytes. The header bytes contain a count field, and descriptor code. The count field indicates the total length of the data block in bytes, thus marking the beginning of the next data block (there are no filler bits). The descriptor code defines: last block in the file (EOF) last block in the record (EOR), restart marker (see the Section on Error Recovery and Restart) or suspect data (i.e., the data being transferred is suspected of errors and is not reliable). This last code is NOT intended for error control within FTP. It is motivated by the desire of sites exchanging certain types of data (e.g., seismic or weather data) to send and receive all the data despite local errors (such as "magnetic tape read errors"), but to indicate in the transmission that certain portions are suspect). Record structures are allowed in this mode, and any representation type may be used.

The header consists of the three bytes. Of the 24 bits of header information, the 16 low order bits shall represent byte count, and the 8 high order bits shall represent descriptor codes as shown below.

Block Header

```
+----------------+----------------+----------------+
| Descriptor     |    Byte Count                   |
|         8 bits |                      16 bits    |
+----------------+----------------+----------------+
```

The descriptor codes are indicated by bit flags in the descriptor byte. Four codes have been assigned, where each code number is the decimal value of the corresponding bit in the byte.

Code	Meaning
128	End of data block is EOR
64	End of data block is EOF
32	Suspected errors in data block
16	Data block is a restart marker

With this encoding, more than one descriptor coded condition may exist for a particular block. As many bits as necessary may be flagged.

The restart marker is embedded in the data stream as an integral number of 8-bit bytes representing printable characters in the language being used over the control connection (e.g., default--NVT-ASCII). <SP> (Space, in the appropriate language) must not be used WITHIN a restart marker.

For example, to transmit a six-character marker, the following would be sent:

```
   +--------+--------+--------+
   |Descrptr|  Byte count     |
   |code= 16|             = 6 |
   +--------+--------+--------+

   +--------+--------+--------+
   | Marker | Marker | Marker |
   | 8 bits | 8 bits | 8 bits |
   +--------+--------+--------+

   +--------+--------+--------+
   | Marker | Marker | Marker |
   | 8 bits | 8 bits | 8 bits |
   +--------+--------+--------+
```

3.4.3. COMPRESSED MODE

There are three kinds of information to be sent: regular data, sent in a byte string; compressed data, consisting of replications or filler; and control information, sent in a two-byte escape sequence. If n>0 bytes (up to 127) of regular data are sent, these n bytes are preceded by a byte with the left-most bit set to 0 and the right-most 7 bits containing the number n.

Byte string:

```
    1       7                8                     8
   +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+
   |0|       n     | |    d(1)       | ... |      d(n)     |
   +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+
                                 ^             ^
                                 |---n bytes---|
                                     of data
```

String of n data bytes d(1),..., d(n)
Count n must be positive.

To compress a string of n replications of the data byte d, the following 2 bytes are sent:

Replicated Byte:

```
     2       6               8
```

```
+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
|1 0|     n     | |       d       |
+-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
```

A string of n filler bytes can be compressed into a single byte, where the filler byte varies with the representation type. If the type is ASCII or EBCDIC the filler byte is <SP> (Space, ASCII code 32, EBCDIC code 64). If the type is Image or Local byte the filler is a zero byte.

Filler String:

```
  2       6
+-+-+-+-+-+-+-+-+
|1 1|     n     |
+-+-+-+-+-+-+-+-+
```

The escape sequence is a double byte, the first of which is the escape byte (all zeros) and the second of which contains descriptor codes as defined in Block mode. The descriptor codes have the same meaning as in Block mode and apply to the succeeding string of bytes.

Compressed mode is useful for obtaining increased bandwidth on very large network transmissions at a little extra CPU cost. It can be most effectively used to reduce the size of printer files such as those generated by RJE hosts.

3.5. ERROR RECOVERY AND RESTART

There is no provision for detecting bits lost or scrambled in data transfer; this level of error control is handled by the TCP. However, a restart procedure is provided to protect users from gross system failures (including failures of a host, an FTP-process, or the underlying network).

The restart procedure is defined only for the block and compressed modes of data transfer. It requires the sender of data to insert a special marker code in the data stream with some marker information. The marker information has meaning only to the sender, but must consist of printable characters in the default or negotiated language of the control connection (ASCII or EBCDIC). The marker could represent a bit-count, a record-count, or any other information by which a system may identify a data checkpoint. The receiver of data, if it implements the restart procedure, would then mark the corresponding position of this marker in the receiving system, and return this information to the user.

In the event of a system failure, the user can restart the data transfer by identifying the marker point with the FTP restart procedure. The following example illustrates the use of the restart procedure.

The sender of the data inserts an appropriate marker block in the data stream at a convenient point. The receiving host marks the corresponding data point in its file system and conveys the last known sender and receiver marker information to the user, either directly or over the control connection in a 110 reply (depending on who is the sender). In the event of a system failure, the user or controller process restarts the server at the last server marker by sending a restart command with server's marker code as its argument. The restart command is transmitted over the control connection and is immediately followed by the command (such as RETR, STOR or LIST) which was being executed when the system failure occurred.

4. FILE TRANSFER FUNCTIONS

The communication channel from the user-PI to the server-PI is established as a TCP connection from the user to the standard server port. The user protocol interpreter is responsible for sending FTP commands and interpreting the replies received; the server-PI interprets commands, sends replies and directs its DTP to set up the data connection and transfer the data. If the second party to the data transfer (the passive transfer process) is the user-DTP, then it is governed through the internal protocol of the user-FTP host; if it is a second server-DTP, then it is governed by its PI on command from the user-PI. The FTP replies are discussed in the next section. In the description of a few of the commands in this section, it is helpful to be explicit about the possible replies.

4.1. FTP COMMANDS

4.1.1. ACCESS CONTROL COMMANDS

The following commands specify access control identifiers (command codes are shown in parentheses).

USER NAME (USER)

The argument field is a Telnet string identifying the user. The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the control connections are made (some servers may require this). Additional identification information in the form of a password and/or an account command may also be required by some servers. Servers may allow a new USER command to be entered at any point in order to change the access control and/or accounting information. This has the effect of flushing any user, password, and account information already supplied and beginning the login sequence again. All transfer parameters are unchanged and any file transfer in progress is completed under the old access control parameters.

PASSWORD (PASS)

The argument field is a Telnet string specifying the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress typeout. It appears that the server has no foolproof way to achieve this. It is therefore the responsibility of the user-FTP process to hide the sensitive password information.

ACCOUNT (ACCT)

The argument field is a Telnet string identifying the user's account. The command is not necessarily related to the USER command, as some sites may require an account for login and others only for specific access, such as storing files. In the latter case the command may arrive at any time.

There are reply codes to differentiate these cases for the automation: when account information is required for login, the response to a successful PASSword command is reply code 332. On the other hand, if account information is NOT required for login, the reply to a successful PASSword command is 230; and if the account information is needed for a command issued later in the dialogue, the server should return a 332 or 532 reply depending on whether it stores (pending receipt of the ACCounT command) or discards the command, respectively.

CHANGE WORKING DIRECTORY (CWD)

This command 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.

CHANGE TO PARENT DIRECTORY (CDUP)

This command 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. See Appendix II for further details.

STRUCTURE MOUNT (SMNT)

This command allows the user to mount a different file system data structure 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.

REINITIALIZE (REIN)

This command terminates a USER, flushing all I/O and account information, except to allow any transfer in progress to be completed. All parameters are reset to the default settings and the control connection is left open. This is identical to the state in which a user finds himself immediately after the control connection is opened. A USER command may be expected to follow.

LOGOUT (QUIT)

This command terminates a USER and if file transfer is not in progress, the server closes the control connection. If file transfer is in progress, the connection will remain open for result response and the server will then close it. If the user-process is transferring files for several USERs but does not wish to close and then reopen connections for each, then the REIN command should be used instead of QUIT.

An unexpected close on the control connection will cause the server to take the effective action of an abort (ABOR) and a logout (QUIT).

4.1.2. TRANSFER PARAMETER COMMANDS

All data transfer parameters have default values, and the commands specifying data transfer parameters are required only if the default parameter values are to be changed. The default value is the last specified value, or if no value has been specified, the standard default value is as stated here. This implies that the server must "remember" the applicable default values. The commands may be in any order except that they must precede the FTP service request. The following commands specify data transfer parameters:

DATA PORT (PORT)

The argument is a HOST-PORT specification for the data port to be used in data connection. There are defaults for both the user and server data ports, and under normal circumstances this command and its reply are not needed. If this command is used, the argument is the concatenation of a 32-bit internet host address and a 16-bit TCP port address. This address information is broken into 8-bit fields and the value of each field is transmitted as a decimal number (in character string representation). The fields are separated by commas. A port command would be:

```
PORT h1,h2,h3,h4,p1,p2
```

where h1 is the high order 8 bits of the internet host address.

PASSIVE (PASV)

This command requests the server-DTP to "listen" on a data port (which is not its default data port) and to wait for a connection rather than initiate one upon receipt of a transfer command. The response to this command includes the host and port address this server is listening on.

REPRESENTATION TYPE (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 Bytesize. The parameters are separated by a <SP> (Space, ASCII code 32).

The following codes are assigned for type:

```
             \    /
A - ASCII |    | N - Non-print
             |-><-| T - Telnet format effectors
E - EBCDIC|    | C - Carriage Control (ASA)
             /    \
I - Image

L <byte size> - Local byte Byte size
```

The default representation type is ASCII Non-print. If the Format parameter is changed, and later just the first argument is changed, Format then returns to the Non-print default.

FILE STRUCTURE (STRU)

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.

TRANSFER MODE (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.

4.1.3. FTP SERVICE COMMANDS

The FTP service commands define the file transfer or the file system function requested by the user. The argument of an FTP service command will normally be a pathname. The syntax of pathnames must conform to server site conventions (with standard defaults applicable), and the language conventions of the control connection. The suggested default handling is to use the last specified device, directory or file name, or the standard default defined for local users. The commands may be in any order except that a "rename from" command must be followed by a "rename to" command and the restart command must be followed by the interrupted service command (e.g., STOR or RETR). The data, when transferred in response to FTP service

commands, shall always be sent over the data connection, except for certain informative replies. The following commands specify FTP service requests:

RETRIEVE (RETR)

This command causes the server-DTP to transfer a copy of the file, specified in the pathname, to the server- or user-DTP at the other end of the data connection. The status and contents of the file at the server site shall be unaffected.

STORE (STOR)

This command causes the server-DTP to accept the data transferred via the data connection and to store the data as a file at the server site. If the file specified in the pathname exists at the server site, then its contents shall be replaced by the data being transferred. A new file is created at the server site if the file specified in the pathname does not already exist.

STORE UNIQUE (STOU)

This command behaves like STOR except that the resultant file is to be created in the current directory under a name unique to that directory. The 250 Transfer Started response must include the name generated.

APPEND (with create) (APPE)

This command causes the server-DTP to accept the data transferred via the data connection and to store the data in a file at the server site. If the file specified in the pathname exists at the server site, then the data shall be appended to that file; otherwise the file specified in the pathname shall be created at the server site.

ALLOCATE (ALLO)

This command may be required by some servers to reserve sufficient storage to accommodate the new file to be transferred. The argument shall be a decimal integer representing the number of bytes (using the logical byte size) of storage to be reserved for the file. For files sent with record or page structure a maximum record or page size (in logical bytes) might also be necessary; this is indicated by a decimal integer in a second argument field of

the command. This second argument is optional, but when present should be separated from the first by the three Telnet characters <SP> R <SP>. This command shall be followed by a STORe or APPEnd command. The ALLO command should be treated as a NOOP (no operation) by those servers which do not require that the maximum size of the file be declared beforehand, and those servers interested in only the maximum record or page size should accept a dummy value in the first argument and ignore it.

RESTART (REST)

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.

RENAME FROM (RNFR)

This command specifies the old pathname of the file which is to be renamed. This command must be immediately followed by a "rename to" command specifying the new file pathname.

RENAME TO (RNTO)

This command specifies the new pathname of the file specified in the immediately preceding "rename from" command. Together the two commands cause a file to be renamed.

ABORT (ABOR)

This command tells the server to abort the previous FTP service command and any associated transfer of data. The abort command may require "special action", as discussed in the Section on FTP Commands, to force recognition by the server. No action is to be taken if the previous command has been completed (including data transfer). The control connection is not to be closed by the server, but the data connection must be closed.

There are two cases for the server upon receipt of this command: (1) the FTP service command was already completed, or (2) the FTP service command is still in progress.

In the first case, the server closes the data connection (if it is open) and responds with a 226 reply, indicating that the abort command was successfully processed.

In the second case, the server aborts the FTP service in progress and closes the data connection, returning a 426 reply to indicate that the service request terminated abnormally. The server then sends a 226 reply, indicating that the abort command was successfully processed.

DELETE (DELE)

This command 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.

REMOVE DIRECTORY (RMD)

This command 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). See Appendix II.

MAKE DIRECTORY (MKD)

This command 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). See Appendix II.

PRINT WORKING DIRECTORY (PWD)

This command causes the name of the current working directory to be returned in the reply. See Appendix II.

LIST (LIST)

This command 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.

NAME LIST (NLST)

This command causes a directory listing to be sent from server to user site. The pathname should specify a directory or other system-specific file group descriptor; a null argument implies the current directory. The server will return a stream of names of files and no other information. The data will be transferred in ASCII or EBCDIC type over the data connection as valid pathname strings separated by <CRLF> or <NL>. (Again the user must ensure that the TYPE is correct.) This command is intended to return information that can be used by a program to further process the files automatically. For example, in the implementation of a "multiple get" function.

SITE PARAMETERS (SITE)

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 SITE command.

SYSTEM (SYST)

This command is used to find out the type of operating system at the server. The reply shall have as its first word one of the system names listed in the current version of the Assigned Numbers document [4].

STATUS (STAT)

This command shall cause a status response to be sent over the control connection in the form of a reply. The command may be sent during a file transfer (along with the Telnet IP and Synch signals--see the Section on FTP Commands) in which case the server will respond with the status of the operation in progress, or it may be sent between file transfers. In the latter case, the command may have an argument field. If the argument is a pathname, the command is analogous to the "list" command except that data shall be transferred over the control connection. If a partial pathname is given, the server may respond with a list of file names or attributes associated with that specification. If no argument is given, the server should return general status information about the server FTP process. This should include current values of all transfer parameters and the status of connections.

HELP (HELP)

This command 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. The reply is type 211 or 214. 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.

NOOP (NOOP)

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

The File Transfer Protocol follows the specifications of the Telnet protocol for all communications over the control connection. Since the language used for Telnet communication may be a negotiated option, all references in the next two sections will be to the "Telnet language" and the corresponding "Telnet end-of-line code". Currently, one may take these to mean NVT-ASCII and <CRLF>. No other specifications of the Telnet protocol will be cited.

FTP commands are "Telnet strings" terminated by the "Telnet end of line code". The command codes themselves are alphabetic characters terminated by the character <SP> (Space) if parameters follow and Telnet-EOL otherwise. The command codes and the semantics of commands are described in this section; the detailed syntax of commands is specified in the Section on Commands, the reply sequences are discussed in the Section on Sequencing of Commands and Replies, and scenarios illustrating the use of commands are provided in the Section on Typical FTP Scenarios.

FTP commands may be partitioned as those specifying access-control identifiers, data transfer parameters, or FTP service requests. Certain commands (such as ABOR, STAT, QUIT) may be sent over the control connection while a data transfer is in progress. Some servers may not be able to monitor the control and data connections simultaneously, in which case some special action will be necessary to get the server's attention. The following ordered format is tentatively recommended:

1. User system inserts the Telnet "Interrupt Process" (IP) signal in the Telnet stream.

2. User system sends the Telnet "Synch" signal.

3. User system inserts the command (e.g., ABOR) in the Telnet stream.

4. Server PI, after receiving "IP", scans the Telnet stream for EXACTLY ONE FTP command.

(For other servers this may not be necessary but the actions listed above should have no unusual effect.)

4.2. FTP REPLIES

Replies to File Transfer Protocol commands are devised to ensure the synchronization of requests and actions in the process of file transfer, and to guarantee that the user process always knows the state of the Server. Every command must generate at least one reply, although there may be more than one; in the latter case, the multiple replies must be easily distinguished. In addition, some commands occur in sequential groups, such as USER, PASS and ACCT, or RNFR and RNTO. The replies show the existence of an intermediate state if all preceding commands have been successful. A failure at any point in the sequence necessitates the repetition of the entire sequence from the beginning.

The details of the command-reply sequence are made explicit in a set of state diagrams below.

An FTP reply consists of a three digit number (transmitted as three alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is intended for the human user. It is intended that the three digits contain enough encoded information that the user-process (the User-PI) will not need to examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be server-dependent, so there are likely to be varying texts for each reply code.

A reply is defined to contain the 3-digit code, followed by Space <SP>, followed by one line of text (where some maximum line length has been specified), and terminated by the Telnet end-of-line code. There will be cases however, where the text is longer than a single line. In these cases the complete text must be bracketed so the User-process knows when it may stop reading the reply (i.e. stop processing input on the control connection) and go do other things. This requires a special format on the first line to indicate that more than one line is coming, and another on the last line to designate it as the last. At least one of these must contain the appropriate reply code to indicate the state of the transaction. To satisfy all factions, it was decided that both the first and last line codes should be the same.

Thus the format for multi-line replies is that the first line will begin with the exact required reply code, followed immediately by a Hyphen, "-" (also known as Minus), followed by text. The last line will begin with the same code, followed immediately by Space <SP>, optionally some text, and the Telnet end-of-line code.

For example:

```
123-First line
Second line
  234 A line beginning with numbers
123 The last line
```

The user-process then simply needs to search for the second occurrence of the same reply code, followed by <SP> (Space), at the beginning of a line, and ignore all intermediary lines. If an intermediary line begins with a 3-digit number, the Server must pad the front to avoid confusion.

This scheme allows standard system routines to be used for reply information (such as for the STAT reply), with "artificial" first and last lines tacked on. In rare cases where these routines are able to generate three digits and a Space at the beginning of any line, the beginning of each text line should be offset by some neutral text, like Space.

This scheme assumes that multi-line replies may not be nested.

The three digits of the reply each have a special significance. This is intended to allow a range of very simple to very sophisticated responses by the user-process. The first digit denotes whether the response is good, bad or incomplete. (Referring to the state diagram), an unsophisticated user-process will be able to determine its next action (proceed as planned, redo, retrench, etc.) by simply examining this first digit. A user-process that wants to know approximately what kind of error occurred (e.g. file system error, command syntax error) may examine the second digit, reserving the third digit for the finest gradation of information (e.g., RNTO command without a preceding RNFR).

There are five values for the first digit of the reply code:

1yz   Positive Preliminary reply

The requested action is being initiated; expect another reply before proceeding with a new command. (The user-process sending another command before the completion reply would be in violation of protocol; but server-FTP processes should queue any commands that arrive while a preceding command is in progress.) This type of reply can be used to indicate that the command was accepted and the user-process may now pay attention to the data connections, for implementations where simultaneous monitoring is difficult. The server-FTP process may send at most, one 1yz reply per command.

2yz   Positive Completion reply

The requested action has been successfully completed. A new request may be initiated.

3yz   Positive Intermediate reply

The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The user should send another command specifying this information. This reply is used in command sequence groups.

4yz   Transient Negative Completion reply

The command was not accepted and the requested action did not take place, but the error condition is temporary and the action may be requested again. The user should return to the beginning of the command sequence, if any. It is difficult to assign a meaning to "transient", particularly when two distinct sites (Server- and User-processes) have to agree on the interpretation. Each reply in the 4yz category might have a slightly different time value, but the intent is that the user-process is encouraged to try again. A rule of thumb in determining if a reply fits into the 4yz or the 5yz (Permanent Negative) category is that replies are 4yz if the commands can be repeated without any change in command form or in properties of the User or Server (e.g., the command is spelled the same with the same arguments used; the user does not change his file access or user name; the server does not put up a new implementation.)

5yz   Permanent Negative Completion reply

The command was not accepted and the requested action did not take place. The User-process is discouraged from repeating the exact request (in the same sequence). Even some "permanent" error conditions can be corrected, so the human user may want to direct his User-process to reinitiate the command sequence by direct action at some point in the future (e.g., after the spelling has been changed, or the user has altered his directory status.)

The following function groupings are encoded in the second digit:

x0z Syntax - These replies refer to syntax errors, syntactically correct commands that don't fit any functional category, unimplemented or superfluous commands.

x1z Information - These are replies to requests for information, such as status or help.

x2z Connections - Replies referring to the control and data connections.

x3z Authentication and accounting - Replies for the login process and accounting procedures.

x4z Unspecified as yet.

x5z File system - These replies indicate the status of the Server file system vis-a-vis the requested transfer or other file system action.

The third digit gives a finer gradation of meaning in each of the function categories, specified by the second digit. The list of replies below will illustrate this. Note that the text associated with each reply is recommended, rather than mandatory, and may even change according to the command with which it is associated. The reply codes, on the other hand, must strictly follow the specifications in the last section; that is, Server implementations should not invent new codes for situations that are only slightly different from the ones described here, but rather should adapt codes already defined.

A command such as TYPE or ALLO whose successful execution does not offer the user-process any new information will cause a 200 reply to be returned. If the command is not implemented by a particular Server-FTP process because it has no relevance to that computer system, for example ALLO at a TOPS20 site, a Positive Completion reply is still desired so that the simple User-process knows it can proceed with its course of action. A 202 reply is used in this case with, for example, the reply text: "No storage allocation necessary." If, on the other hand, the command requests a non-site-specific action and is unimplemented, the response is 502. A refinement of that is the 504 reply for a command that is implemented, but that requests an unimplemented parameter.

4.2.1 Reply Codes by Function Groups

200 Command okay.

500 Syntax error, command unrecognized.
This may include errors such as command line too long.

501 Syntax error in parameters or arguments.

202 Command not implemented, superfluous at this site.

502 Command not implemented.

503 Bad sequence of commands.

504 Command not implemented for that parameter.

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 "=").

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.

    120 Service ready in nnn minutes.
    220 Service ready for new user.
    221 Service closing control connection.
        Logged out if appropriate.
    421 Service not available, closing control connection.
        This may be a reply to any command if the service knows it
        must shut down.
    125 Data connection already open; transfer starting.
    225 Data connection open; no transfer in progress.
    425 Can't open data connection.
    226 Closing data connection.
        Requested file action successful (for example, file
        transfer or file abort).
    426 Connection closed; transfer aborted.
    227 Entering Passive Mode (h1,h2,h3,h4,p1,p2).

    230 User logged in, proceed.
    530 Not logged in.
    331 User name okay, need password.
    332 Need account for login.
    532 Need account for storing files.
    150 File status okay; about to open data connection.
    250 Requested file action okay, completed.
    257 "PATHNAME" created.
    350 Requested file action pending further information.
    450 Requested file action not taken.
        File unavailable (e.g., file busy).
    550 Requested action not taken.
        File unavailable (e.g., file not found, no access).
    451 Requested action aborted. Local error in processing.
    551 Requested action aborted. Page type unknown.
    452 Requested action not taken.
        Insufficient storage space in system.
    552 Requested file action aborted.
        Exceeded storage allocation (for current directory or
        dataset).
    553 Requested action not taken.
        File name not allowed.
```

4.2.2 Numeric Order List of Reply Codes

```
    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.
452 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.

5. DECLARATIVE SPECIFICATIONS

5.1. MINIMUM IMPLEMENTATION

In order to make FTP workable without needless error messages, the following minimum implementation is required for all servers:

TYPE - ASCII Non-print
MODE - Stream
STRUCTURE - File, Record
COMMANDS - USER, QUIT, PORT,
TYPE, MODE, STRU,
for the default values
RETR, STOR,
NOOP.

The default values for transfer parameters are:

TYPE - ASCII Non-print
MODE - Stream
STRU - File

All hosts must accept the above as the standard defaults.

5.2. CONNECTIONS

The server protocol interpreter shall "listen" on Port L. The user or user protocol interpreter shall initiate the full-duplex control connection. Server- and user- processes should follow the conventions of the Telnet protocol as specified in the ARPA-Internet Protocol Handbook [1]. Servers are under no obligation to provide for editing of command lines and may require that it be done in the user host. The control connection shall be closed by the server at the user's request after all transfers and replies are completed.

The user-DTP must "listen" on the specified data port; this may be the default user port (U) or a port specified in the PORT command. The server shall initiate the data connection from his own default data port (L-1) using the specified user data port. The direction of the transfer and the port used will be determined by the FTP service command.

Note that all FTP implementation must support data transfer using the default port, and that only the USER-PI may initiate the use of non-default ports.

When data is to be transferred between two servers, A and B (refer to Figure 2), the user-PI, C, sets up control connections with both server-PI's. One of the servers, say A, is then sent a PASV command telling him to "listen" on his data port rather than initiate a connection when he receives a transfer service command. When the user-PI receives an acknowledgment to the PASV command, which includes the identity of the host and port being listened on, the user-PI then sends A's port, a, to B in a PORT command; a reply is returned. The user-PI may then send the corresponding service commands to A and B. Server B initiates the connection and the transfer proceeds. The command-reply sequence is listed below where the messages are vertically synchronous but horizontally asynchronous:

```
User-PI - Server A              User-PI - Server B
------------------              ------------------

C->A : Connect                  C->B : Connect
C->A : PASV
A->C : 227 Entering Passive Mode. A1,A2,A3,A4,a1,a2
                                C->B : PORT A1,A2,A3,A4,a1,a2
                                B->C : 200 Okay
C->A : STOR                     C->B : RETR
           B->A : Connect to HOST-A, PORT-a
```

Figure 3

The data connection shall be closed by the server under the conditions described in the Section on Establishing Data Connections. If the data connection is to be closed following a data transfer where closing the connection is not required to indicate the end-of-file, the server must do so immediately. Waiting until after a new transfer command is not permitted because the user-process will have already tested the data connection to see if it needs to do a "listen"; (remember that the user must "listen" on a closed data port BEFORE sending the transfer request). To prevent a race condition here, the server sends a reply (226) after closing the data connection (or if the connection is left open, a "file transfer completed" reply (250) and the user-PI should wait for one of these replies before issuing a new transfer command).

Any time either the user or server see that the connection is being closed by the other side, it should promptly read any remaining data queued on the connection and issue the close on its own side.

5.3. COMMANDS

The commands are Telnet character strings transmitted over the control connections as described in the Section on FTP Commands. The command functions and semantics are described in the Section on Access Control Commands, Transfer Parameter Commands, FTP Service Commands, and Miscellaneous Commands. The command syntax is specified here.

The commands begin with a command code followed by an argument field. The command codes are four or fewer alphabetic characters. Upper and lower case alphabetic characters are to be treated identically. Thus, any of the following may represent the retrieve command:

```
            RETR    Retr    retr    ReTr    rETr
```

This also applies to any symbols representing parameter values, such as A or a for ASCII TYPE. The command codes and the argument fields are separated by one or more spaces.

The argument field consists of a variable length character string ending with the character sequence <CRLF> (Carriage Return, Line Feed) for NVT-ASCII representation; for other negotiated languages a different end of line character might be used. It should be noted that the server is to take no action until the end of line code is received.

The syntax is specified below in NVT-ASCII. All characters in the argument field are ASCII characters including any ASCII represented decimal integers. Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied.

5.3.1. FTP COMMANDS

The following are the FTP commands:

```
      USER <SP> <username> <CRLF>
      PASS <SP> <password> <CRLF>
      ACCT <SP> <account-information> <CRLF>
      CWD  <SP> <pathname> <CRLF>
      CDUP <CRLF>
      SMNT <SP> <pathname> <CRLF>
      QUIT <CRLF>
      REIN <CRLF>
      PORT <SP> <host-port> <CRLF>
      PASV <CRLF>
      TYPE <SP> <type-code> <CRLF>
      STRU <SP> <structure-code> <CRLF>
      MODE <SP> <mode-code> <CRLF>
      RETR <SP> <pathname> <CRLF>
      STOR <SP> <pathname> <CRLF>
      STOU <CRLF>
      APPE <SP> <pathname> <CRLF>
      ALLO <SP> <decimal-integer>
          [<SP> R <SP> <decimal-integer>] <CRLF>
      REST <SP> <marker> <CRLF>
      RNFR <SP> <pathname> <CRLF>
      RNTO <SP> <pathname> <CRLF>
      ABOR <CRLF>
      DELE <SP> <pathname> <CRLF>
      RMD  <SP> <pathname> <CRLF>
      MKD  <SP> <pathname> <CRLF>
      PWD  <CRLF>
      LIST [<SP> <pathname>] <CRLF>
      NLST [<SP> <pathname>] <CRLF>
      SITE <SP> <string> <CRLF>
      SYST <CRLF>
      STAT [<SP> <pathname>] <CRLF>
      HELP [<SP> <string>] <CRLF>
      NOOP <CRLF>
```

5.3.2. FTP COMMAND ARGUMENTS

The syntax of the above argument fields (using BNF notation where applicable) is:

```
<username> ::= <string>
<password> ::= <string>
<account-information> ::= <string>
<string> ::= <char> | <char><string>
<char> ::= any of the 128 ASCII characters except <CR> and
<LF>
<marker> ::= <pr-string>
<pr-string> ::= <pr-char> | <pr-char><pr-string>
<pr-char> ::= printable characters, any
              ASCII code 33 through 126
<byte-size> ::= <number>
<host-port> ::= <host-number>,<port-number>
<host-number> ::= <number>,<number>,<number>,<number>
<port-number> ::= <number>,<number>
<number> ::= any decimal integer 1 through 255
<form-code> ::= N | T | C
<type-code> ::= A [<sp> <form-code>]
              | E [<sp> <form-code>]
              | I
              | L <sp> <byte-size>
<structure-code> ::= F | R | P
<mode-code> ::= S | B | C
<pathname> ::= <string>
<decimal-integer> ::= any decimal integer
```

5.4. SEQUENCING OF COMMANDS AND REPLIES

The communication between the user and server is intended to be an alternating dialogue. As such, the user issues an FTP command and the server responds with a prompt primary reply. The user should wait for this initial primary success or failure response before sending further commands.

Certain commands require a second reply for which the user should also wait. These replies may, for example, report on the progress or completion of file transfer or the closing of the data connection. They are secondary replies to file transfer commands.

One important group of informational replies is the connection greetings. Under normal circumstances, a server will send a 220 reply, "awaiting input", when the connection is completed. The user should wait for this greeting message before sending any commands. If the server is unable to accept input right away, a 120 "expected delay" reply should be sent immediately and a 220 reply when ready. The user will then know not to hang up if there is a delay.

Spontaneous Replies

Sometimes "the system" spontaneously has a message to be sent to a user (usually all users). For example, "System going down in 15 minutes". There is no provision in FTP for such spontaneous information to be sent from the server to the user. It is recommended that such information be queued in the server-PI and delivered to the user-PI in the next reply (possibly making it a multi-line reply).

The table below lists alternative success and failure replies for each command. These must be strictly adhered to; a server may substitute text in the replies, but the meaning and action implied by the code numbers and by the specific command reply sequence cannot be altered.

Command-Reply Sequences

In this section, the command-reply sequence is presented. Each command is listed with its possible replies; command groups are listed together. Preliminary replies are listed first (with their succeeding replies indented and under them), then positive and negative completion, and finally intermediary replies with the remaining commands from the sequence following. This listing forms the basis for the state diagrams, which will be presented separately.

```
Connection Establishment
   120
      220
   220
   421
Login
   USER
      230
      530
      500, 501, 421
      331, 332
   PASS
      230
      202
      530
      500, 501, 503, 421
      332
   ACCT
      230
      202
      530
      500, 501, 503, 421
   CWD
      250
      500, 501, 502, 421, 530, 550
   CDUP
      200
      500, 501, 502, 421, 530, 550
   SMNT
      202, 250
      500, 501, 502, 421, 530, 550
Logout
   REIN
      120
         220
      220
      421
      500, 502
   QUIT
      221
      500

Transfer parameters
   PORT
      200
      500, 501, 421, 530
   PASV
      227
      500, 501, 502, 421, 530
   MODE
      200
      500, 501, 504, 421, 530
   TYPE
      200
      500, 501, 504, 421, 530
   STRU
      200
      500, 501, 504, 421, 530

File action commands
```

```
ALLO
   200
   202
   500, 501, 504, 421, 530
REST
   500, 501, 502, 421, 530
   350
STOR
   125, 150
      (110)
      226, 250
      425, 426, 451, 551, 552
   532, 450, 452, 553
   500, 501, 421, 530
STOU
   125, 150
      (110)
      226, 250
      425, 426, 451, 551, 552
   532, 450, 452, 553
   500, 501, 421, 530
RETR
   125, 150
      (110)
      226, 250
      425, 426, 451
   450, 550
   500, 501, 421, 530

LIST
   125, 150
      226, 250
      425, 426, 451
   450
   500, 501, 502, 421, 530
NLST
   125, 150
      226, 250
      425, 426, 451
   450
   500, 501, 502, 421, 530
APPE
   125, 150
      (110)
      226, 250
      425, 426, 451, 551, 552
   532, 450, 550, 452, 553
   500, 501, 502, 421, 530
RNFR
   450, 550
   500, 501, 502, 421, 530
   350
RNTO
   250
   532, 553
   500, 501, 502, 503, 421, 530
DELE
   250
   450, 550
   500, 501, 502, 421, 530
RMD
   250
   500, 501, 502, 421, 530, 550
MKD
   257
   500, 501, 502, 421, 530, 550
PWD
   257
   500, 501, 502, 421, 550
ABOR
   225, 226
```

```
            500, 501, 502, 421

      Informational commands

         SYST
            215
            500, 501, 502, 421
         STAT
            211, 212, 213
            450
            500, 501, 502, 421, 530
         HELP
            211, 214
            500, 501, 502, 421

      Miscellaneous commands

         SITE
            200
            202
            500, 501, 530
         NOOP
            200
            500 421
```

6. STATE DIAGRAMS

Here we present state diagrams for a very simple minded FTP implementation. Only the first digit of the reply codes is used. There is one state diagram for each group of FTP commands or command sequences.

The command groupings were determined by constructing a model for each command then collecting together the commands with structurally identical models.

For each command or command sequence there are three possible outcomes: success (S), failure (F), and error (E). In the state diagrams below we use the symbol B for "begin", and the symbol W for "wait for reply".

We first present the diagram that represents the largest group of FTP commands:

```
                           1,3    +---+
                      ----------->| E |
                     |            +---+
                     |
   +---+    cmd    +---+    2      +---+
   | B |---------->| W |---------->| S |
   +---+           +---+           +---+
                     |
                     |     4,5    +---+
                      ----------->| F |
                                  +---+
```

This diagram models the commands:

ABOR, ALLO, DELE, CWD, CDUP, SMNT, HELP, MODE, NOOP, PASV, QUIT, SITE, PORT, SYST, STAT, RMD, MKD, PWD, STRU, and TYPE.

The other large group of commands is represented by a very similar diagram:

```
                            3      +---+
                       ----------->| E |
                      |            +---+
                      |
   +---+    cmd    +---+    2      +---+
   | B |---------->| W |---------->| S |
   +---+       --->+---+           +---+
              |     | |
              |     | |     4,5    +---+
              |  1  |  ----------->| F |
               -----               +---+
```

   This diagram models the commands:

      APPE, LIST, NLST, REIN, RETR, STOR, and STOU.

Note that this second model could also be used to represent the first group of commands, the only difference being that in the first group the 100 series replies are unexpected and therefore treated as error, while the second group expects (some may require) 100 series replies. Remember that at most, one 100 series reply is allowed per command.

The remaining diagrams model command sequences, perhaps the simplest of these is the rename sequence:

```
   +---+   RNFR    +---+    1,2    +---+
   | B |---------->| W |---------->| E |
   +---+           +---+        -->+---+
                    | |        |
             3      | | 4,5    |
      --------------  ------   |
     |                      |  |   +---+
     |               ------------->| S |
     |              |   1,3 |  |   +---+
     |             2|  --------
     |              | |     |
     V              | |     |
   +---+   RNTO    +---+ 4,5 ----->+---+
   |   |---------->| W |---------->| F |
   +---+           +---+           +---+
```

The next diagram is a simple model of the Restart command:

```
   +---+   REST    +---+    1,2    +---+
   | B |---------->| W |---------->| E |
   +---+           +---+        -->+---+
                    | |        |
             3      | | 4,5    |
      --------------  ------   |
     |                      |  |   +---+
     |               ------------->| S |
     |              |   3   |  |   +---+
     |             2|  --------
     |              | |     |
     V              | |     |
   +---+   cmd     +---+ 4,5 ----->+---+
   |   |---------->| W |---------->| F |
   +---+        -->+---+           +---+
               |      |
               |  1   |
                ------
```

      Where "cmd" is APPE, STOR, or RETR.

We note that the above three models are similar. The Restart differs from the Rename two only in the treatment of 100 series replies at the second stage, while the second group expects (some may require)

100 series replies. Remember that at most, one 100 series reply is allowed per command.

The most complicated diagram is for the Login sequence:

```
                            1
      +---+   USER    +---+------------->+---+
      | B |---------->| W | 2       ---->| E |
      +---+           +---+------  |  -->+---+
                       | |       | | |
                     3 | | 4,5   | | |
         --------------   -----  | | |
        |                      | | | |
        |                      | | | |
        |                 ---------  |
        |               1|     | |   |
        V                |     | |   |
      +---+   PASS    +---+ 2  |  ------>+---+
      |   |---------->| W |------------->| S |
      +---+           +---+   ---------->+---+
                       | |   | |     |
                     3 | |4,5| |     |
         --------------   --------   |
        |                    | |  |  |
        |                    | |  |  |
        |                 -----------
        |             1,3|   | |  |
        V                |  2| |  |
      +---+   ACCT    +---+--  |   ----->+---+
      |   |---------->| W | 4,5 -------->| F |
      +---+           +---+------------->+---+
```

Finally, we present a generalized diagram that could be used to model the command and reply interchange:

```
               ------------------------------------
              |                                    |
      Begin   |                                    |
        |     V                                    |
        |   +---+  cmd   +---+ 2         +---+     |
         -->|   |------->|   |---------->|   |     |
            |   |        | W |           | S |-----|
         -->|   |     -->|   |-----      |   |     |
        |   +---+    |   +---+ 4,5 |     +---+     |
        |     |      |    | |      |               |
        |     |      |   1| |3     |     +---+     |
        |     |      |    | |      |     |   |     |
        |     |       ----  |       ---->| F |-----
        |     |             |            |   |
        |     |             |            +---+
         -------------------
              |
              |
              V
             End
```

7. TYPICAL FTP SCENARIO

User at host U wanting to transfer files to/from host S:

In general, the user will communicate to the server via a mediating user-FTP process. The following may be a typical scenario. The user-FTP prompts are shown in parentheses, '---->' represents commands from host U to host S, and '<----' represents replies from host S to host U.

```
   LOCAL COMMANDS BY USER              ACTION INVOLVED

   ftp (host) multics<CR>         Connect to host S, port L,
                                  establishing control connections.
                                  <---- 220 Service ready <CRLF>.
   username Doe <CR>              USER Doe<CRLF>---->
                                  <---- 331 User name ok,
                                            need password<CRLF>.
   password mumble <CR>           PASS mumble<CRLF>---->
                                  <---- 230 User logged in<CRLF>.
   retrieve (local type) ASCII<CR>
   (local pathname) test 1 <CR>   User-FTP opens local file in ASCII.
   (for. pathname) test.pl1<CR>   RETR test.pl1<CRLF> ---->
                                  <---- 150 File status okay;
                                        about to open data
                                        connection<CRLF>.
                                  Server makes data connection
                                  to port U.

                                  <---- 226 Closing data connection,
                                      file transfer successful<CRLF>.
   type Image<CR>                 TYPE I<CRLF> ---->
                                  <---- 200 Command OK<CRLF>
   store (local type) image<CR>
   (local pathname) file dump<CR> User-FTP opens local file in Image.
   (for.pathname) >udd>cn>fd<CR>  STOR >udd>cn>fd<CRLF> ---->
                                  <---- 550 Access denied<CRLF>
   terminate                      QUIT <CRLF> ---->
                                  Server closes all
                                  connections.
```

8. CONNECTION ESTABLISHMENT

The FTP control connection is established via TCP between the user process port U and the server process port L. This protocol is assigned the service port 21 (25 octal), that is L=21.

APPENDIX I - PAGE STRUCTURE

The need for FTP to support page structure derives principally from the need to support efficient transmission of files between TOPS-20 systems, particularly the files used by NLS.

The file system of TOPS-20 is based on the concept of pages. The operating system is most efficient at manipulating files as pages. The operating system provides an interface to the file system so that many applications view files as sequential streams of characters. However, a few applications use the underlying page structures directly, and some of these create holey files.

A TOPS-20 disk file consists of four things: a pathname, a page table, a (possibly empty) set of pages, and a set of attributes.

The pathname is specified in the RETR or STOR command. It includes the directory name, file name, file name extension, and generation number.

The page table contains up to 2**18 entries. Each entry may be EMPTY, or may point to a page. If it is not empty, there are also some page-specific access bits; not all pages of a file need have the same access protection.

A page is a contiguous set of 512 words of 36 bits each.

The attributes of the file, in the File Descriptor Block (FDB), contain such things as creation time, write time, read time, writer's byte-size, end-of-file pointer, count of reads and writes, backup system tape numbers, etc.

Note that there is NO requirement that entries in the page table be contiguous. There may be empty page table slots between occupied ones. Also, the end of file pointer is simply a number. There is no requirement that it in fact point at the "last" datum in the file. Ordinary sequential I/O calls in TOPS-20 will cause the end of file pointer to be left after the last datum written, but other operations may cause it not to be so, if a particular programming system so requires.

In fact, in both of these special cases, "holey" files and end-of-file pointers NOT at the end of the file, occur with NLS data files.

The TOPS-20 paged files can be sent with the FTP transfer parameters: TYPE L 36, STRU P, and MODE S (in fact, any mode could be used).

Each page of information has a header. Each header field, which is a logical byte, is a TOPS-20 word, since the TYPE is L 36.

The header fields are:

Word 0: Header Length.

The header length is 5.

Word 1: Page Index.

If the data is a disk file page, this is the number of that page in the file's page map. Empty pages (holes) in the file are simply not sent. Note that a hole is NOT the same as a page of zeros.

Word 2: Data Length.

The number of data words in this page, following the header. Thus, the total length of the transmission unit is the Header Length plus the Data Length.

Word 3: Page Type.

A code for what type of chunk this is. A data page is type 3, the FDB page is type 2.

Word 4: Page Access Control.

The access bits associated with the page in the file's page map. (This full word quantity is put into AC2 of an SPACS by the program reading from net to disk.)

After the header are Data Length data words. Data Length is currently either 512 for a data page or 31 for an FDB. Trailing zeros in a disk file page may be discarded, making Data Length less than 512 in that case.

APPENDIX II - DIRECTORY COMMANDS

Since UNIX has a tree-like directory structure in which directories are as easy to manipulate as ordinary files, it is useful to expand the FTP servers on these machines to include commands which deal with the creation of directories. Since there are other hosts on the ARPA-Internet which have tree-like directories (including TOPS-20 and Multics), these commands are as general as possible.

Four directory commands have been added to FTP:

MKD pathname

Make a directory with the name "pathname".

RMD pathname

Remove the directory with the name "pathname".

PWD

Print the current working directory name.

CDUP

Change to the parent of the current working directory.

The "pathname" argument should be created (removed) as a subdirectory of the current working directory, unless the "pathname" string contains sufficient information to specify otherwise to the server, e.g., "pathname" is an absolute pathname (in UNIX and Multics), or pathname is something like "<abso.lute.path>" to TOPS-20.

REPLY CODES

The CDUP command 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 for CDUP be identical to the reply codes of CWD.

The reply codes for RMD be identical to the reply codes for its file analogue, DELE.

The reply codes for MKD, however, are a bit more complicated. A freshly created directory will probably be the object of a future

CWD command. Unfortunately, the argument to MKD may not always be a suitable argument for CWD. This is the case, for example, when a TOPS-20 subdirectory is created by giving just the subdirectory name. That is, with a TOPS-20 server FTP, the command sequence

```
MKD MYDIR
CWD MYDIR
```

will fail. The new directory may only be referred to by its "absolute" name; e.g., if the MKD command above were issued while connected to the directory <DFRANKLIN>, the new subdirectory could only be referred to by the name <DFRANKLIN.MYDIR>.

Even on UNIX and Multics, however, the argument given to MKD may not be suitable. If it is a "relative" pathname (i.e., a pathname which is interpreted relative to the current directory), the user would need to be in the same current directory in order to reach the subdirectory. Depending on the application, this may be inconvenient. It is not very robust in any case.

To solve these problems, upon successful completion of an MKD command, the server should return a line of the form:

```
257<space>"<directory-name>"<space><commentary>
```

That is, the server will tell the user what string to use when referring to the created directory. The directory name can contain any character; embedded double-quotes should be escaped by double-quotes (the "quote-doubling" convention).

For example, a user connects to the directory /usr/dm, and creates a subdirectory, named pathname:

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD pathname
257 "/usr/dm/pathname" directory created
```

An example with an embedded double quote:

```
MKD foo"bar
257 "/usr/dm/foo""bar" directory created
CWD /usr/dm/foo"bar
200 directory changed to /usr/dm/foo"bar
```

The prior existence of a subdirectory with the same name is an error, and the server must return an "access denied" error reply in that case.

```
CWD /usr/dm
200 directory changed to /usr/dm
MKD pathname
521-"/usr/dm/pathname" directory already exists;
521 taking no action.
```

The failure replies for MKD are analogous to its file creating cousin, STOR. Also, an "access denied" return is given if a file name with the same name as the subdirectory will conflict with the creation of the subdirectory (this is a problem on UNIX, but shouldn't be one on TOPS-20).

Essentially because the PWD command returns the same type of information as the successful MKD command, the successful PWD command uses the 257 reply code as well.

SUBTLETIES

Because these commands will be most useful in transferring subtrees from one machine to another, carefully observe that the argument to MKD is to be interpreted as a sub-directory of the current working directory, unless it contains enough information for the destination host to tell otherwise. A hypothetical example of its use in the TOPS-20 world:

```
CWD <some.where>
200 Working directory changed
MKD overrainbow
257 "<some.where.overrainbow>" directory created
CWD overrainbow
431 No such directory
CWD <some.where.overrainbow>
200 Working directory changed

CWD <some.where>
200 Working directory changed to <some.where>
MKD <unambiguous>
257 "<unambiguous>" directory created
CWD <unambiguous>
```

Note that the first example results in a subdirectory of the connected directory. In contrast, the argument in the second example contains enough information for TOPS-20 to tell that the <unambiguous> directory is a top-level directory. Note also that in the first example the user "violated" the protocol by attempting to access the freshly created directory with a name other than the one returned by TOPS-20. Problems could have resulted in this case had there been an <overrainbow> directory; this is an ambiguity inherent in some TOPS-20 implementations. Similar considerations apply to the RMD command. The point is this: except where to do so would violate a host's conventions for denoting relative versus absolute pathnames, the host should treat the operands of the MKD and RMD commands as subdirectories. The 257 reply to the MKD command must always contain the absolute pathname of the created directory.

Bhushan, Abhay, "A File Transfer Protocol", RFC 114 (NIC 5823), MIT-Project MAC, 16 April 1971.

Harslem, Eric, and John Heafner, "Comments on RFC 114 (A File Transfer Protocol)", RFC 141 (NIC 6726), RAND, 29 April 1971.

Bhushan, Abhay, et al, "The File Transfer Protocol", RFC 172 (NIC 6794), MIT-Project MAC, 23 June 1971.

Braden, Bob, "Comments on DTP and FTP Proposals", RFC 238 (NIC 7663), UCLA/CCN, 29 September 1971.

Bhushan, Abhay, et al, "The File Transfer Protocol", RFC 265 (NIC 7813), MIT-Project MAC, 17 November 1971.

McKenzie, Alex, "A Suggested Addition to File Transfer Protocol", RFC 281 (NIC 8163), BBN, 8 December 1971.

Bhushan, Abhay, "The Use of "Set Data Type" Transaction in File Transfer Protocol", RFC 294 (NIC 8304), MIT-Project MAC, 25 January 1972.

Bhushan, Abhay, "The File Transfer Protocol", RFC 354 (NIC 10596), MIT-Project MAC, 8 July 1972.

Bhushan, Abhay, "Comments on the File Transfer Protocol (RFC 354)", RFC 385 (NIC 11357), MIT-Project MAC, 18 August 1972.

Hicks, Greg, "User FTP Documentation", RFC 412 (NIC 12404), Utah, 27 November 1972.

Bhushan, Abhay, "File Transfer Protocol (FTP) Status and Further Comments", RFC 414 (NIC 12406), MIT-Project MAC, 20 November 1972.

Braden, Bob, "Comments on File Transfer Protocol", RFC 430 (NIC 13299), UCLA/CCN, 7 February 1973.

Thomas, Bob, and Bob Clements, "FTP Server-Server Interaction", RFC 438 (NIC 13770), BBN, 15 January 1973.

Braden, Bob, "Print Files in FTP", RFC 448 (NIC 13299), UCLA/CCN, 27 February 1973.

McKenzie, Alex, "File Transfer Protocol", RFC 454 (NIC 14333), BBN, 16 February 1973.

Bressler, Bob, and Bob Thomas, "Mail Retrieval via FTP", RFC 458 (NIC 14378), BBN-NET and BBN-TENEX, 20 February 1973.

Neigus, Nancy, "File Transfer Protocol", RFC 542 (NIC 17759), BBN, 12 July 1973.

Krilanovich, Mark, and George Gregg, "Comments on the File Transfer Protocol", RFC 607 (NIC 21255), UCSB, 7 January 1974.

Pogran, Ken, and Nancy Neigus, "Response to RFC 607 - Comments on the File Transfer Protocol", RFC 614 (NIC 21530), BBN, 28 January 1974.

Krilanovich, Mark, George Gregg, Wayne Hathaway, and Jim White, "Comments on the File Transfer Protocol", RFC 624 (NIC 22054), UCSB, Ames Research Center, SRI-ARC, 28 February 1974.

Bhushan, Abhay, "FTP Comments and Response to RFC 430", RFC 463 (NIC 14573), MIT-DMCG, 21 February 1973.

Braden, Bob, "FTP Data Compression", RFC 468 (NIC 14742), UCLA/CCN, 8 March 1973.

Bhushan, Abhay, "FTP and Network Mail System", RFC 475 (NIC 14919), MIT-DMCG, 6 March 1973.

Bressler, Bob, and Bob Thomas "FTP Server-Server Interaction - II", RFC 478 (NIC 14947), BBN-NET and BBN-TENEX, 26 March 1973.

White, Jim, "Use of FTP by the NIC Journal", RFC 479 (NIC 14948), SRI-ARC, 8 March 1973.

White, Jim, "Host-Dependent FTP Parameters", RFC 480 (NIC 14949), SRI-ARC, 8 March 1973.

Padlipsky, Mike, "An FTP Command-Naming Problem", RFC 506 (NIC 16157), MIT-Multics, 26 June 1973.

Day, John, "Memo to FTP Group (Proposal for File Access Protocol)", RFC 520 (NIC 16819), Illinois, 25 June 1973.

Merryman, Robert, "The UCSD-CC Server-FTP Facility", RFC 532 (NIC 17451), UCSD-CC, 22 June 1973.

Braden, Bob, "TENEX FTP Problem", RFC 571 (NIC 18974), UCLA/CCN, 15 November 1973.

McKenzie, Alex, and Jon Postel, "Telnet and FTP Implementation - Schedule Change", RFC 593 (NIC 20615), BBN and MITRE, 29 November 1973.

Sussman, Julie, "FTP Error Code Usage for More Reliable Mail Service", RFC 630 (NIC 30237), BBN, 10 April 1974.

Postel, Jon, "Revised FTP Reply Codes", RFC 640 (NIC 30843), UCLA/NMC, 5 June 1974.

Harvey, Brian, "Leaving Well Enough Alone", RFC 686 (NIC 32481), SU-AI, 10 May 1975.

Harvey, Brian, "One More Try on the FTP", RFC 691 (NIC 32700), SU-AI, 28 May 1975.

Lieb, J., "CWD Command of FTP", RFC 697 (NIC 32963), 14 July 1975.

Harrenstien, Ken, "FTP Extension: XSEN", RFC 737 (NIC 42217), SRI-KL, 31 October 1977.

Harrenstien, Ken, "FTP Extension: XRSQ/XRCP", RFC 743 (NIC 42758), SRI-KL, 30 December 1977.

Lebling, P. David, "Survey of FTP Mail and MLFL", RFC 751, MIT, 10 December 1978.

Postel, Jon, "File Transfer Protocol Specification", RFC 765, ISI, June 1980.

Mankins, David, Dan Franklin, and Buzz Owen, "Directory Oriented FTP Commands", RFC 776, BBN, December 1980.

Padlipsky, Michael, "FTP Unique-Named Store Command", RFC 949, MITRE, July 1985.

REFERENCES

[1] Feinler, Elizabeth, "Internet Protocol Transition Workbook", Network Information Center, SRI International, March 1982.

[2] Postel, Jon, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[3] Postel, Jon, and Joyce Reynolds, "Telnet Protocol Specification", RFC 854, ISI, May 1983.

[4] Reynolds, Joyce, and Jon Postel, "Assigned Numbers", RFC 943, ISI, April 1985.

RFC 1123 (FTP section) Index

RFC 1123 (FTP section)

Network Working Group   J. Postel (ISI)
Request for Comments: 1121   L. Kleinrock (UCLA)
V. Cerf (NRI)
B. Boehm (UCLA)
September 1989

Act One - The Poems

Status of this Memo

This RFC presents a collection of poems that were presented at "Act One", a symposium held partially in celebration of the 20th anniversary of the ARPANET. Distribution of this memo is unlimited.

4.1 FILE TRANSFER PROTOCOL -- FTP

4.1.1 INTRODUCTION

The File Transfer Protocol FTP is the primary Internet standard for file transfer. The current specification is contained in RFC-959 [FTP:1].

FTP uses separate simultaneous TCP connections for control and for data transfer. The FTP protocol includes many features, some of which are not commonly implemented. However, for every feature in FTP, there exists at least one implementation. The minimum implementation defined in RFC-959 was too small, so a somewhat larger minimum implementation is defined here.

Internet users have been unnecessarily burdened for years by deficient FTP implementations. Protocol implementors have suffered from the erroneous opinion that implementing FTP ought to be a small and trivial task. This is wrong, because FTP has a user interface, because it has to deal (correctly) with the whole variety of communication and operating system errors that may occur, and because it has to handle the great diversity of real file systems in the world.

4.1.2. PROTOCOL WALK-THROUGH

4.1.2.1 LOCAL Type: RFC-959 Section 3.1.1.4

An FTP program MUST support TYPE I ("IMAGE" or binary type) as well as TYPE L 8 ("LOCAL" type with logical byte size 8). A machine whose memory is organized into m-bit words, where m is not a multiple of 8, MAY also support TYPE L m.

DISCUSSION:

The command "TYPE L 8" is often required to transfer binary data between a machine whose memory is organized into (e.g.) 36-bit words and a machine with an 8-bit byte organization. For an 8-bit byte machine, TYPE L 8 is equivalent to IMAGE.

"TYPE L m" is sometimes specified to the FTP programs on two m-bit word machines to ensure the correct transfer of a native-mode binary file from one machine to the other. However, this command should have the same effect on these machines as "TYPE I".

4.1.2.2 Telnet Format Control: RFC-959 Section 3.1.1.5.2

A host that makes no distinction between TYPE N and TYPE T SHOULD implement TYPE T to be identical to TYPE N.

DISCUSSION:

This provision should ease interoperation with hosts that do make this distinction.

Many hosts represent text files internally as strings of ASCII characters, using the embedded ASCII format effector characters (LF, BS, FF, ...) to control the format when a file is printed. For such hosts, there is no distinction between "print" files and other files. However, systems that use record structured files typically need a special format for printable files (e.g., ASA carriage control). For the latter hosts, FTP allows a choice of TYPE N or TYPE T.

4.1.2.3 Page Structure: RFC-959 Section 3.1.2.3 and Appendix I

Implementation of page structure is NOT RECOMMENDED in general. However, if a host system does need to implement FTP for "random access" or "holey" files, it MUST use the defined page structure format rather than define a new private FTP format.

4.1.2.4 Data Structure Transformations: RFC-959 Section 3.1.2

An FTP transformation between record-structure and file-structure SHOULD be invertible, to the extent possible while making the result useful on the target host.

DISCUSSION:

RFC-959 required strict invertibility between record-structure and file-structure, but in practice, efficiency and convenience often preclude it. Therefore, the requirement is being relaxed. There are two different objectives for transferring a file: processing it on the target host, or just storage. For storage, strict invertibility is important. For processing, the file created on the target host needs to be in the format expected by application programs on that host.

As an example of the conflict, imagine a record-oriented operating system that requires some data files to have exactly 80 bytes in each record. While STORing a file on such a host, an FTP Server must be able to pad each line or record to 80 bytes; a later retrieval of such a file cannot be strictly invertible.

4.1.2.5 Data Connection Management: RFC-959 Section 3.3

A User-FTP that uses STREAM mode SHOULD send a PORT command to assign a non-default data port before each transfer command is issued.

DISCUSSION:

This is required because of the long delay after a TCP connection is closed until its socket pair can be reused, to allow multiple transfers during a single FTP session. Sending a port command can avoided if a transfer mode other than stream is used, by leaving the data transfer connection open between transfers.

4.1.2.6 PASV Command: RFC-959 Section 4.1.2

A server-FTP MUST implement the PASV command.

If multiple third-party transfers are to be executed during the same session, a new PASV command MUST be issued before each transfer command, to obtain a unique port pair.

IMPLEMENTATION:

The format of the 227 reply to a PASV command is not well standardized. In particular, an FTP client cannot assume that the parentheses shown on page 40 of RFC-959 will be present (and in fact, Figure 3 on page 43 omits them). Therefore, a User-FTP program that interprets the PASV reply must scan the reply for the first digit of the host and port numbers.

Note that the host number h1,h2,h3,h4 is the IP address of the server host that is sending the reply, and that p1,p2 is a non-default data transfer port that PASV has assigned.

4.1.2.7 LIST and NLST Commands: RFC-959 Section 4.1.3

The data returned by an NLST command MUST contain only a simple list of legal pathnames, such that the server can use them directly as the arguments of subsequent data transfer commands for the individual files.

The data returned by a LIST or NLST command SHOULD use an implied TYPE AN, unless the current type is EBCDIC, in which case an implied TYPE EN SHOULD be used.

DISCUSSION:

Many FTP clients support macro-commands that will get or put files matching a wildcard specification, using NLST to obtain a list of pathnames. The expansion of "multiple-put" is local to the client, but "multiple-get" requires cooperation by the server.

The implied type for LIST and NLST is designed to provide compatibility with existing User-FTPs, and in particular with multiple-get commands.

4.1.2.8 SITE Command: RFC-959 Section 4.1.3

A Server-FTP SHOULD use the SITE command for non-standard features, rather than invent new private commands or unstandardized extensions to existing commands.

4.1.2.9 STOU Command: RFC-959 Section 4.1.3

The STOU command stores into a uniquely named file. When it receives an STOU command, a Server-FTP MUST return the actual file name in the "125 Transfer Starting" or the "150 Opening Data Connection" message that precedes the transfer (the 250 reply code mentioned in RFC-959 is incorrect). The exact format of these messages is hereby defined to be as follows:

```
125 FILE: pppp
150 FILE: pppp
```

where pppp represents the unique pathname of the file that will be written.

4.1.2.10 Telnet End-of-line Code: RFC-959, Page 34

Implementors MUST NOT assume any correspondence between READ boundaries on the control connection and the Telnet EOL sequences (CR LF).

DISCUSSION:

Thus, a server-FTP (or User-FTP) must continue reading characters from the control connection until a complete Telnet EOL sequence is encountered, before processing the command (or response, respectively). Conversely, a single READ from the control connection may include more than one FTP command.

4.1.2.11 FTP Replies: RFC-959 Section 4.2, Page 35

A Server-FTP MUST send only correctly formatted replies on the control connection. Note that RFC-959 (unlike earlier versions of the FTP spec) contains no provision for a "spontaneous" reply message.

A Server-FTP SHOULD use the reply codes defined in RFC-959 whenever they apply. However, a server-FTP MAY use a different reply code when needed, as long as the general rules of Section 4.2 are followed. When the implementor has a choice between a 4xx and 5xx reply code, a Server-FTP SHOULD send a 4xx (temporary failure) code when there is any reasonable possibility that a failed FTP will succeed a few hours later.

A User-FTP SHOULD generally use only the highest-order digit of a 3-digit reply code for making a procedural decision, to prevent difficulties when a Server-FTP uses non-standard reply codes.

A User-FTP MUST be able to handle multi-line replies. If the implementation imposes a limit on the number of lines and if this limit is exceeded, the User-FTP MUST recover, e.g., by ignoring the excess lines until the end of the multi-line reply is reached.

A User-FTP SHOULD NOT interpret a 421 reply code ("Service not available, closing control connection") specially, but SHOULD detect closing of the control connection by the server.

DISCUSSION:

Server implementations that fail to strictly follow the reply rules often cause FTP user programs to hang. Note that RFC-959 resolved ambiguities in the reply rules found in earlier FTP specifications and must be followed.

It is important to choose FTP reply codes that properly distinguish between temporary and permanent failures, to allow the successful use of file transfer client daemons. These programs depend on the reply codes to decide whether or not to retry a failed transfer; using a permanent failure code (5xx) for a temporary error will cause these programs to give up unnecessarily. When the meaning of a reply matches exactly the text shown in RFC-959, uniformity will be enhanced by using the RFC-959 text verbatim. However, a Server-FTP implementor is encouraged to choose reply text that conveys specific system-dependent information, when appropriate.

4.1.2.12 Connections: RFC-959 Section 5.2

The words "and the port used" in the second paragraph of this section of RFC-959 are erroneous (historical), and they should be ignored.

On a multihomed server host, the default data transfer port (L-1) MUST be associated with the same local IP address as the corresponding control connection to port L.

A user-FTP MUST NOT send any Telnet controls other than SYNCH and IP on an FTP control connection. In particular, it MUST NOT attempt to negotiate Telnet options on the control connection. However, a server-FTP MUST be capable of accepting and refusing Telnet negotiations (i.e., sending DONT/WONT).

DISCUSSION:

Although the RFC says: "Server- and User- processes should follow the conventions for the Telnet protocol...[on the control connection]", it is not the intent that Telnet option negotiation is to be employed.

4.1.2.13 Minimum Implementation; RFC-959 Section 5.1

The following commands and options MUST be supported by every server-FTP and user-FTP, except in cases where the underlying file system or operating system does not allow or support a particular command.

```
     Type: ASCII Non-print, IMAGE, LOCAL 8
     Mode: Stream
     Structure: File, Record*
     Commands:
        USER, PASS, ACCT,
        PORT, PASV,
        TYPE, MODE, STRU,
        RETR, STOR, APPE,
        RNFR, RNTO, DELE,
        CWD,  CDUP, RMD,  MKD,  PWD,
        LIST, NLST,
        SYST, STAT,
        HELP, NOOP, QUIT.
```

*Record structure is REQUIRED only for hosts whose file systems support record structure.

DISCUSSION:
Vendors are encouraged to implement a larger subset of the protocol. For example, there are important robustness features in the protocol (e.g., Restart, ABOR, block mode) that would be an aid to some Internet users but are not widely implemented.

A host that does not have record structures in its file system may still accept files with STRU R, recording the byte stream literally.

4.1.3 SPECIFIC ISSUES

4.1.3.1 Non-standard Command Verbs

FTP allows "experimental" commands, whose names begin with "X". If these commands are subsequently adopted as standards, there may still be existing implementations using the "X" form. At present, this is true for the directory commands:

RFC-959	"Experimental"
* MKD	XMKD
* RMD	XRMD
* PWD	XPWD
* CDUP	XCUP
* CWD	XCWD

All FTP implementations SHOULD recognize both forms of these commands, by simply equating them with extra entries in the command lookup table.

IMPLEMENTATION:
A User-FTP can access a server that supports only the "X" forms by implementing a mode switch, or automatically using the following procedure: if the RFC-959 form of one of the above commands is rejected with a 500 or 502 response code, then try the experimental form; any other response would be passed to the user.

4.1.3.2 Idle Timeout

A Server-FTP process SHOULD have an idle timeout, which will terminate the process and close the control connection if the server is inactive (i.e., no command or data transfer in progress) for a long period of time. The idle timeout time SHOULD be configurable, and the default should be at least 5 minutes.

A client FTP process ("User-PI" in RFC-959) will need timeouts on responses only if it is invoked from a program.

DISCUSSION:

Without a timeout, a Server-FTP process may be left pending indefinitely if the corresponding client crashes without closing the control connection.

4.1.3.3 Concurrency of Data and Control

DISCUSSION:

The intent of the designers of FTP was that a user should be able to send a STAT command at any time while data transfer was in progress and that the server-FTP would reply immediately with status -- e.g., the number of bytes transferred so far. Similarly, an ABOR command should be possible at any time during a data transfer.

Unfortunately, some small-machine operating systems make such concurrent programming difficult, and some other implementers seek minimal solutions, so some FTP implementations do not allow concurrent use of the data and control connections. Even such a minimal server must be prepared to accept and defer a STAT or ABOR command that arrives during data transfer.

4.1.3.4 FTP Restart Mechanism

The description of the 110 reply on pp. 40-41 of RFC-959 is incorrect; the correct description is as follows. A restart reply message, sent over the control connection from the receiving FTP to the User-FTP, has the format:

110 MARK ssss = rrrr

Here:

* ssss is a text string that appeared in a Restart Marker in the data stream and encodes a position in the sender's file system;

* rrrr encodes the corresponding position in the receiver's file system.

The encoding, which is specific to a particular file system and network implementation, is always generated and interpreted by the same system, either sender or receiver.

When an FTP that implements restart receives a Restart Marker in the data stream, it SHOULD force the data to that point to be written to stable storage before encoding the corresponding position rrrr. An FTP sending Restart Markers MUST NOT assume that 110 replies will be returned synchronously with the data, i.e., it must not await a 110 reply before sending more data.

Two new reply codes are hereby defined for errors encountered in restarting a transfer:

554 Requested action not taken: invalid REST parameter.

A 554 reply may result from a FTP service command that follows a REST command. The reply indicates that the existing file at the Server-FTP cannot be repositioned as specified in the REST.

555 Requested action not taken: type or stru mismatch.

A 555 reply may result from an APPE command or from any FTP service command following a REST command. The reply indicates that there is some mismatch between the current transfer parameters (type and stru) and the attributes of the existing file.

DISCUSSION:

Note that the FTP Restart mechanism requires that Block or Compressed mode be used for data transfer, to allow the Restart Markers to be included within the data stream. The frequency of Restart Markers can be low.

Restart Markers mark a place in the data stream, but the receiver may be performing some transformation on the data as it is stored into stable storage. In general, the receiver's encoding must include any state information necessary to restart this transformation at any point of the FTP data stream. For example, in TYPE A transfers, some receiver hosts transform CR LF sequences into a single LF character on disk. If a Restart Marker happens to fall between CR and LF, the receiver must encode in rrrr that the transfer must be restarted in a "CR has been seen and discarded" state.

Note that the Restart Marker is required to be encoded as a string of printable ASCII characters, regardless of the type of the data.

RFC-959 says that restart information is to be returned "to the user". This should not be taken literally. In general, the User-FTP should save the restart information (ssss,rrrr) in stable storage, e.g., append it to a restart control file. An empty restart control file should be created when the transfer first starts and deleted automatically when the transfer completes successfully. It is suggested that this file have a name derived in an easily-identifiable manner from the name of the file being transferred and the remote host name; this is analogous to the means used by many text editors for naming "backup" files.

There are three cases for FTP restart.

(1) User-to-Server Transfer

The User-FTP puts Restart Markers <ssss> at convenient places in the data stream. When the Server-FTP receives a Marker, it writes all prior data to disk, encodes its file system position and transformation state as rrrr, and returns a "110 MARK ssss = rrrr" reply over the control connection. The User-FTP appends the pair (ssss,rrrr) to its restart control file.

To restart the transfer, the User-FTP fetches the last (ssss,rrrr) pair from the restart control file, repositions its local file system and transformation state using ssss, and sends the command "REST rrrr" to the Server-FTP.

(2) Server-to-User Transfer

The Server-FTP puts Restart Markers <ssss> at convenient places in the data stream. When the User-FTP receives a Marker, it writes all prior data to disk, encodes its file system position and transformation state as rrrr, and appends the pair (rrrr,ssss) to its restart control file.

To restart the transfer, the User-FTP fetches the last (rrrr,ssss) pair from the restart control file, repositions its local file system and transformation state using rrrr, and sends the command "REST ssss" to the Server-FTP.

(3) Server-to-Server ("Third-Party") Transfer

The sending Server-FTP puts Restart Markers <ssss> at convenient places in the data stream. When it receives a Marker, the receiving Server-FTP writes all prior data to disk, encodes its file system position and transformation state as rrrr, and sends a "110 MARK ssss = rrrr" reply over the control connection to the User. The User-FTP appends the pair (ssss,rrrr) to its restart control file.

To restart the transfer, the User-FTP fetches the last (ssss,rrrr) pair from the restart control file, sends "REST ssss" to the sending Server-FTP, and sends "REST rrrr" to the receiving Server-FTP.

4.1.4 FTP/USER INTERFACE

This section discusses the user interface for a User-FTP program.

4.1.4.1 Pathname Specification

Since FTP is intended for use in a heterogeneous environment, User-FTP implementations MUST support remote pathnames as arbitrary character strings, so that their form and content are not limited by the conventions of the local operating system.

DISCUSSION:

In particular, remote pathnames can be of arbitrary length, and all the printing ASCII characters as well as space (0x20) must be allowed. RFC-959 allows a pathname to contain any 7-bit ASCII character except CR or LF.

4.1.4.2 "QUOTE" Command

A User-FTP program MUST implement a "QUOTE" command that will pass an arbitrary character string to the server and display all resulting response messages to the user.

To make the "QUOTE" command useful, a User-FTP SHOULD send transfer control commands to the server as the user enters them, rather than saving all the commands and sending them to the server only when a data transfer is started.

DISCUSSION:

The "QUOTE" command is essential to allow the user to access servers that require system-specific commands (e.g., SITE or ALLO), or to invoke new or optional features that are not implemented by the User-FTP. For example, "QUOTE" may be used to specify "TYPE A T" to send a print file to hosts that require the distinction, even if the User-FTP does not recognize that TYPE.

4.1.4.3 Displaying Replies to User

A User-FTP SHOULD display to the user the full text of all error reply messages it receives. It SHOULD have a "verbose" mode in which all commands it sends and the full text and reply codes it receives are displayed, for diagnosis of problems.

4.1.4.4 Maintaining Synchronization

The state machine in a User-FTP SHOULD be forgiving of missing and unexpected reply messages, in order to maintain command synchronization with the server.

4.1.5   FTP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Implement TYPE T if same as TYPE N	4.1.2.2		x				
File/Record transform invertible if poss.	4.1.2.4		x				
User-FTP send PORT cmd for stream mode	4.1.2.5		x				
Server-FTP implement PASV	4.1.2.6	x					
PASV is per-transfer	4.1.2.6	x					
NLST reply usable in RETR cmds	4.1.2.7	x					
Implied type for LIST and NLST	4.1.2.7		x				
SITE cmd for non-standard features	4.1.2.8		x				
STOU cmd return pathname as specified	4.1.2.9	x					
Use TCP READ boundaries on control conn.	4.1.2.10					x	
Server-FTP send only correct reply format	4.1.2.11	x					
Server-FTP use defined reply code if poss.	4.1.2.11		x				
New reply code following Section 4.2	4.1.2.11			x			
User-FTP use only high digit of reply	4.1.2.11		x				
User-FTP handle multi-line reply lines	4.1.2.11	x					
User-FTP handle 421 reply specially	4.1.2.11				x		
Default data port same IP addr as ctl conn	4.1.2.12	x					
User-FTP send Telnet cmds exc. SYNCH, IP	4.1.2.12					x	
User-FTP negotiate Telnet options	4.1.2.12					x	
Server-FTP handle Telnet options	4.1.2.12	x					
Handle "Experimental" directory cmds	4.1.3.1		x				
Idle timeout in server-FTP	4.1.3.2		x				
Configurable idle timeout	4.1.3.2		x				
Receiver checkpoint data at Restart Marker	4.1.3.4		x				
Sender assume 110 replies are synchronous	4.1.3.4					x	
Support TYPE:							
ASCII - Non-Print (AN)	4.1.2.13	x					
ASCII - Telnet (AT) -- if same as AN	4.1.2.2		x				
ASCII - Carriage Control (AC)	959 3.1.1.5.2			x			
EBCDIC - (any form)	959 3.1.1.2			x			
IMAGE	4.1.2.1	x					
LOCAL 8	4.1.2.1	x					
LOCAL m	4.1.2.1			x			2
Support MODE:							
Stream	4.1.2.13	x					
Block	959 3.4.2			x			
Support STRUCTURE:							
File	4.1.2.13	x					

Record	4.1.2.13	x					3
Page	4.1.2.3				x		
Support commands:							
USER	4.1.2.13	x					
PASS	4.1.2.13	x					
ACCT	4.1.2.13	x					
CWD	4.1.2.13	x					
CDUP	4.1.2.13	x					
SMNT	959 5.3.1			x			
REIN	959 5.3.1			x			
QUIT	4.1.2.13	x					
PORT	4.1.2.13	x					
PASV	4.1.2.6	x					
TYPE	4.1.2.13	x					1
STRU	4.1.2.13	x					1
MODE	4.1.2.13	x					1
RETR	4.1.2.13	x					
STOR	4.1.2.13	x					
STOU	959 5.3.1			x			
APPE	4.1.2.13	x					
ALLO	959 5.3.1			x			
REST	959 5.3.1			x			
RNFR	4.1.2.13	x					
RNTO	4.1.2.13	x					
ABOR	959 5.3.1			x			
DELE	4.1.2.13	x					
RMD	4.1.2.13	x					
MKD	4.1.2.13	x					
PWD	4.1.2.13	x					
LIST	4.1.2.13	x					
NLST	4.1.2.13	x					
SITE	4.1.2.8			x			
STAT	4.1.2.13	x					
SYST	4.1.2.13	x					
HELP	4.1.2.13	x					
NOOP	4.1.2.13	x					
User Interface:							
Arbitrary pathnames	4.1.4.1	x					
Implement "QUOTE" command	4.1.4.2	x					
Transfer control commands immediately	4.1.4.2		x				
Display error messages to user	4.1.4.3		x				
Verbose mode	4.1.4.3		x				
Maintain synchronization with server	4.1.4.4		x				

Footnotes:

(1) For the values shown earlier.

(2) Here m is number of bits in a memory word.

(3) Required for host with record-structured file system, optional otherwise.

UnZip

About

Unzip.exe is included with the War FTP Daemon in order to provide a simple method to CRC check incomming .zip achives. Unzip.exe is freeware and part of the unz520xN.exe distribution.

Original documentation

UNZIP(1L) MISC. REFERENCE MANUAL PAGES UNZIP(1L)

NAME

unzip - list, test and extract compressed files in a ZIP archive

SYNOPSIS

unzip [-Z] [-cflptuvz[abjnoqsCLMVX$]] file[.zip] [file(s) ...] [-x xfile(s) ...] [-d exdir]

DESCRIPTION

unzip will list, test, or extract files from a ZIP archive, commonly found on MS-DOS systems. The default behavior (with no options) is to extract into the current directory (and subdirectories below it) all files from the specified ZIP archive. A companion program, zip(1L), creates ZIP archives; both programs are compatible with archives created by PKWARE's PKZIP and PKUNZIP for MS-DOS, but in many cases the program options or default behaviors differ.

ARGUMENTS

file[.zip]

Path of the ZIP archive(s). If the file specification is a wildcard, each matching file is processed in an order determined by the operating system (or file system). Only the filename can be a wildcard; the path itself cannot. Wildcard expressions are similar to Unix egrep(1) (regular) expressions and may contain:

* matches a sequence of 0 or more characters

? matches exactly 1 character

[...]

matches any single character found inside the brackets; ranges are specified by a beginning character, a hyphen, and an ending character. If an exclamation point or a caret (`!' or `^') follows the left bracket, then the range of characters within the brackets is complemented (that is, anything except the characters inside the brackets is considered a match).

(Be sure to quote any character that might otherwise be interpreted or modified by the operating system, particularly under Unix and VMS.) If no matches are found, the specification is assumed to be a literal filename; and if that also fails, the suffix .zip is appended. Note that self-extracting ZIP files are supported, as with any other ZIP archive; just specify the .exe suffix (if any) explicitly.

[file(s)]

An optional list of archive members to be processed,

separated by spaces. (VMS versions compiled with VMSCLI defined must delimit files with commas instead. See -v in OPTIONS below.) Regular expressions (wildcards) may be used to match multiple members; see above. Again, be sure to quote expressions that would otherwise be expanded or modified by the operating system.

[-x xfile(s)]

An optional list of archive members to be excluded from processing. Since wildcard characters match directory separators (`/'), this option may be used to exclude any files that are in subdirectories. For example, ``unzip foo *.[ch] -x */*" would extract all C source files in the main directory, but none in any subdirectories. Without the -x option, all C source files in all directories within the zipfile would be extracted.

[-d exdir]

An optional directory to which to extract files. By default, all files and subdirectories are recreated in the current directory; the -d option allows extraction in an arbitrary directory (always assuming one has permission to write to the directory). This option need not appear at the end of the command line; it is also accepted before the zipfile specification (with the normal options), immediately after the zipfile specification, or between the file(s) and the -x option. The option and directory may be concatenated without any white space between them, but note that this may cause normal shell behavior to be suppressed. In particular, ``-d ~" (tilde) is expanded by Unix C shells into the name of the user's home directory, but ``-d~" is treated as a literal subdirectory ``~" of the current directory.

OPTIONS

Note that, in order to support obsolescent hardware, unzip's usage screen is limited to 22 or 23 lines and should therefore be considered only a reminder of the basic unzip syntax rather than an exhaustive list of all possible flags.

-Z zipinfo(1L) mode. If the first option on the command line is -Z, the remaining options are taken to be zipinfo(1L) options. See the appropriate manual page for a description of these options.

-c extract files to stdout/screen (``CRT''). This option is similar to the -p option except that the name of each file is printed as it is extracted, the -a option is allowed, and ASCII-EBCDIC conversion is automatically performed if appropriate. This option is not listed in the unzip usage screen.

-f freshen existing files, i.e., extract only those files that already exist on disk and that are newer than the disk copies. By default unzip queries before overwriting, but the -o option may be used to suppress the queries. Note that under many operating systems, the TZ (timezone) environment variable must be set correctly in order for -f and -u to work properly (under Unix the variable is usually set automatically). The reasons for this are somewhat subtle but have to do with the differences between DOS-format file times (always local time) and Unix-format times (always in GMT) and the necessity to compare the two. A typical TZ value is ``PST8PDT'' (US Pacific time with automatic adjustment for Daylight Savings Time or ``summer time'').

-l list archive files (short format). The names, uncompressed file sizes and modification dates and times of the specified files are printed, along with totals for all files specified. If UnZip was compiled with OS2_EAS defined, the -l option also lists columns for the sizes of stored OS/2 extended attributes (EAs) and OS/2 access control lists (ACLs). In addition, the zipfile comment and individual file comments (if any) are displayed. If a file was archived from a single-case file system (for example, the old MS-DOS FAT file system) and the -L option was given, the filename is converted to lowercase and is prefixed with a caret (^).

-p extract files to pipe (stdout). Nothing but the file data is sent to stdout, and the files are always extracted in binary format, just as they are stored (no conversions).

-t test archive files. This option extracts each specified file in memory and compares the CRC (cyclic redundancy check, an enhanced checksum) of the expanded file with the original file's stored CRC value.

-u update existing files and create new ones if needed. This option performs the same function as the -f option, extracting (with query) files that are newer than those with the same name on disk, and in addition it extracts those files that do not already exist on disk. See -f above for information on setting the timezone properly.

-v be verbose or print diagnostic version info. This option has evolved and now behaves as both an option and a modifier. As an option it has two purposes: when a zipfile is specified with no other options, -v lists archive files verbosely, adding to the basic -l info the compression method, compressed size, compression ratio and 32-bit CRC. When no zipfile is specified (that is, the complete command is simply ``unzip -v''), a diagnostic screen is printed. In addition to the normal header with release date and version, unzip lists the home Info-ZIP ftp site and where to find a list of other ftp and non-ftp sites; the target operating system for which it was compiled, as well as (possibly) the hardware on which it was compiled, the compiler and version used, and the compilation date; any special compilation options that might affect the program's operation (see also DECRYPTION below); and any options stored in environment variables that might do the same (see ENVIRONMENT OPTIONS below). As a modifier it works in conjunction with other options (e.g., -t) to produce more verbose or debugging output; this is not yet fully implemented but will be in future releases.

-z display only the archive comment.

MODIFIERS

-a convert text files. Ordinarily all files are extracted exactly as they are stored (as ``binary'' files). The -a option causes files identified by zip as text files (those with the `t' label in zipinfo listings, rather than `b') to be automatically extracted as such, converting line endings, end-of-file characters and the character set itself as necessary. (For example, Unix files use line feeds (LFs) for end-of-line (EOL) and have no end-of-file (EOF) marker; Macintoshes use carriage returns (CRs) for EOLs; and most PC operating systems use CR+LF for EOLs and control-Z for EOF. In addition, IBM mainframes and the Michigan Terminal System use EBCDIC rather than the more common ASCII character set, and NT supports Unicode.) Note that zip's identification of text files is by no means perfect; some ``text'' files may actually be binary and vice versa. unzip therefore prints ``[text]'' or ``[binary]'' as a visual check for each file it extracts when using the -a option. The -aa option forces all files to be extracted as text, regardless of the supposed file type.

-b treat all files as binary (no text conversions). This is a shortcut for ---a.

-C match filenames case-insensitively. unzip's philosophy is ``you get what you ask for'' (this is also

responsible for the -L/-U change; see the relevant options below). Because some file systems are fully case-sensitive (notably those under the Unix operating system) and because both ZIP archives and unzip itself are portable across platforms, unzip's default behavior is to match both wildcard and literal filenames case-sensitively. That is, specifying ``makefile'' on the command line will only match ``makefile'' in the archive, not ``Makefile'' or ``MAKEFILE'' (and similarly for wildcard specifications). Since this does not correspond to the behavior of many other operating/file systems (for example, OS/2 HPFS, which preserves mixed case but is not sensitive to it), the -C option may be used to force all filename matches to be case-insensitive. In the example above, all three files would then match ``makefile'' (or ``make*'', or similar). The -C option affects files in both the normal file list and the excluded-file list (xlist).

-j junk paths. The archive's directory structure is not recreated; all files are deposited in the extraction directory (by default, the current one).

-L convert to lowercase any filename originating on an uppercase-only operating system or file system. (This was unzip's default behavior in releases prior to 5.11; the new default behavior is identical to the old behavior with the -U option, which is now obsolete and will be removed in a future release.) Depending on the archiver, files archived under single-case file systems (VMS, old MS-DOS FAT, etc.) may be stored as all-uppercase names; this can be ugly or inconvenient when extracting to a case-preserving file system such as OS/2 HPFS or a case-sensitive one such as under Unix. By default unzip lists and extracts such filenames exactly as they're stored (excepting truncation, conversion of unsupported characters, etc.); this option causes the names of all files from certain systems to be converted to lowercase.

-M pipe all output through an internal pager similar to the Unixmore(1) command. At the end of a screenful of output, unzip pauses with a ``--More--'' prompt; the next screenful may be viewed by pressing the Enter (Return) key or the space bar. unzip can be terminated by pressing the ``q'' key and, on some systems, the Enter/Return key. Unlike Unix more(1), there is no forward-searching or editing capability. Also, unzip doesn't notice if long lines wrap at the edge of the screen, effectively resulting in the printing of two or more lines and the likelihood that some text will scroll off the top of the screen before being viewed.

On some systems the number of available lines on the screen is not detected, in which case unzip assumes the height is 24 lines.

-n never overwrite existing files. If a file already exists, skip the extraction of that file without prompting. By default unzip queries before extracting any file that already exists; the user may choose to overwrite only the current file, overwrite all files, skip extraction of the current file, skip extraction of all existing files, or rename the current file.

-o overwrite existing files without prompting. This is a dangerous option, so use it with care. (It is often used with -f, however.)

-q perform operations quietly (-qq = even quieter). Ordinarily unzip prints the names of the files it's extracting or testing, the extraction methods, any file or zipfile comments that may be stored in the archive, and possibly a summary when finished with each archive. The -q[q] options suppress the printing of some or all of these messages.

-s [OS/2, NT, MS-DOS] convert spaces in filenames to underscores. Since all PC operating systems allow spaces in filenames, unzip by default extracts filenames with spaces intact (e.g., ``EA DATA. SF''). This can be awkward, however, since MS-DOS in particular does not gracefully support spaces in filenames. Conversion of spaces to underscores can eliminate the awkwardness in some cases.

-U (obsolete; to be removed in a future release) leave filenames uppercase if created under MS-DOS, VMS, etc. See -L above.

-V retain (VMS) file version numbers. VMS files can be stored with a version number, in the format file.ext;##. By default the ``;##'' version numbers are stripped, but this option allows them to be retained. (On file systems that limit filenames to particularly short lengths, the version numbers may be truncated or stripped regardless of this option.)

-X [VMS, Unix, OS/2] restore owner/protection info (UICs) under VMS, or user and group info (UID/GID) under Unix, or access control lists (ACLs) under certain network-enabled versions of OS/2 (Warp Server with IBM LAN Server/Requester 3.0 to 5.0; Warp Connect with IBM Peer 1.0). In most cases this will require special system privileges; but under Unix, for example, a user who

belongs to several groups can restore files owned by any of those groups, as long as the user IDs match his or her own. Note that ordinary file attributes are always restored; this option applies only to optional, extra ownership info available on some operating systems. [Note that NT's access control lists are probably compatible with OS/2's. A future release will support cross-platform storage and restoration of ACLs.]

-$ [MS-DOS, OS/2, NT] restore the volume label if the extraction medium is removable (e.g., a diskette). Doubling the option (-$$) allows fixed media (hard disks) to be labelled as well. By default, volume labels are ignored.

ENVIRONMENT OPTIONS

unzip's default behavior may be modified via options placed in an environment variable. This can be done with any option, but it is probably most useful with the -a, -L, -C, -q, -o, or -n modifiers: make unzip auto-convert text files by default, make it convert filenames from uppercase systems to lowercase, make it match names case-insensitively, make it quieter, or make it always overwrite or never overwrite files as it extracts them. For example, to make unzip act as quietly as possible, only reporting errors, one would use one of the following commands:

```
UNZIP=-qq; export UNZIP      Unix Bourne shell
setenv UNZIP -qq             Unix C shell
set UNZIP=-qq                OS/2 or MS-DOS
define UNZIP_OPTS "-qq"      VMS (quotes for lowercase)
```

Environment options are, in effect, considered to be just like any other command-line options, except that they are effectively the first options on the command line. To override an environment option, one may use the ``minus operator'' to remove it. For instance, to override one of the quiet-flags in the example above, use the command

```
     unzip --q[other options] zipfile
```

The first hyphen is the normal switch character, and the second is a minus sign, acting on the q option. Thus the effect here is to cancel one quantum of quietness. To cancel both quiet flags, two (or more) minuses may be used:

```
     unzip -t--q zipfile
     unzip ---qt zipfile
```

(the two are equivalent). This may seem awkward or confusing, but it is reasonably intuitive: just ignore the first hyphen and go from there. It is also consistent with the behavior of Unix nice(1).

As suggested by the examples above, the default variable names are UNZIP_OPTS for VMS (where the symbol used to install unzip as a foreign command would otherwise be confused with the environment variable), and UNZIP for all other operating systems. For compatibility with zip(1L), UNZIPOPT is also accepted (don't ask). If both UNZIP and UNZIPOPT are defined, however, UNZIP takes precedence. unzip's diagnostic option (-v with no zipfile name) can be used to check the values of all four possible unzip and zipinfo environment variables.

The timezone variable (TZ) should be set according to the local timezone in order for the -f and -u to operate correctly. See the description of -f above for details. This variable may also be necessary in order for timestamps on extracted files to be set correctly.

DECRYPTION

Encrypted archives are fully supported by Info-ZIP software, but due to United States export restrictions, the encryption and decryption sources are not packaged with the regular unzip and zip distributions. Since the crypt sources were written by Europeans, however, they are freely available at sites throughout the world; see the file ``Where'' in any Info-ZIP source or binary distribution for locations both inside and outside the US.

Because of the separate distribution, not all compiled versions of unzip support decryption. To check a version for crypt support, either attempt to test or extract an encrypted archive, or else check unzip's diagnostic screen (see the -v option above) for ``[decryption]'' as one of the special compilation options.

There are no runtime options for decryption; if a zipfile member is encrypted, unzip will prompt for the password without echoing what is typed. unzip continues to use the same password as long as it appears to be valid; it does this by testing a 12-byte header. The correct password will always check out against the header, but there is a 1-in-256 chance that an incorrect password will as well. (This is a security feature of the PKWARE zipfile format; it helps prevent brute-force attacks that might otherwise gain a large speed advantage by testing only the header.) In the case that an incorrect password is given but it passes the header test anyway, either an incorrect CRC will be generated for the extracted data or else unzip will fail during the extraction because the ``decrypted'' bytes do not constitute a valid compressed data stream.

If the first password fails the header check on some file, unzip will prompt for another password, and so on until all files are extracted. If a password is not known, entering a null password (that is, just a carriage return) is taken as a signal to skip all further prompting. Only unencrypted files in the archive(s) will thereafter be extracted. (Actually that's not quite true; older versions of zip(1L) and zipcloak(1L) allowed null passwords, so unzip checks each encrypted file to see if the null password works. This may result in ``false positives'' and extraction errors, as noted above.)

Note that there is presently no way to avoid interactive decryption. This is another security feature: plaintext passwords given on the command line or stored in files constitute a risk because they may be seen by others. Future releases may (under protest, with great disapproval) support such shenanigans.

EXAMPLES

To use unzip to extract all members of the archive letters.zip into the current directory and subdirectories below it, creating any subdirectories as necessary:

```
unzip letters
```

To extract all members of letters.zip into the current directory only:

```
unzip -j letters
```

To test letters.zip, printing only a summary message indicating whether the archive is OK or not:

```
unzip -tq letters
```

To test all zipfiles in the current directory, printing only the summaries:

```
unzip -tq \*.zip
```

(The backslash before the asterisk is only required if the shell expands wildcards, as in Unix; double quotes could have been used instead, as in the source examples below.) To extract to standard output all members of letters.zip whose names end in .tex, auto-converting to the local end-of-line convention and piping the output into more(1):

```
unzip -ca letters \*.tex | more
```

To extract the binary file paper1.dvi to standard output and pipe it to a printing program:

```
unzip -p articles paper1.dvi | dvips
```

To extract all FORTRAN and C source files--*.f, *.c, *.h, and Makefile--into the /tmp directory:

```
unzip source.zip "*.[fch]" Makefile -d /tmp
```

(the double quotes are necessary only in Unix and only if globbing is turned on). To extract all FORTRAN and C source files, regardless of case (e.g., both *.c and *.C, and any makefile, Makefile, MAKEFILE or similar):

```
unzip -C source.zip "*.[fch]" makefile -d /tmp
```

To extract any such files but convert any uppercase MS-DOS or VMS names to lowercase and convert the line-endings of all of the files to the local standard (without respect to any files that might be marked ``binary"):

```
unzip -aaCL source.zip "*.[fch]" makefile -d /tmp
```

To extract only newer versions of the files already in the current directory, without querying (NOTE: be careful of unzipping in one timezone a zipfile created in another--ZIP archives to date contain no timezone information, and a ``newer" file from an eastern timezone may, in fact, be older):

```
unzip -fo sources
```

To extract newer versions of the files already in the current directory and to create any files not already there (same caveat as previous example):

```
unzip -uo sources
```

To display a diagnostic screen showing which unzip and zipinfo options are stored in environment variables, whether decryption support was compiled in, the compiler with which unzip was compiled, etc.:

```
unzip -v
```

In the last five examples, assume that UNZIP or UNZIP_OPTS is set to -q. To do a singly quiet listing:

```
unzip -l file.zip
```

To do a doubly quiet listing:

```
unzip -ql file.zip
```

(Note that the ``.zip'' is generally not necessary.) To do a standard listing:

```
     unzip --ql file.zip
or
     unzip -l-q file.zip
or
     unzip -l--q file.zip          (extra minuses don't hurt)
```

TIPS

The current maintainer, being a lazy sort, finds it very useful to define a pair of aliases: tt for ``unzip -tq'' and ii for ``unzip -Z'' (or ``zipinfo''). One may then simply type ``tt zipfile'' to test an archive, something that is worth making a habit of doing. With luck unzip will report ``No errors detected in compressed data of zipfile.zip,'' after which one may breathe a sigh of relief.

The maintainer also finds it useful to set the UNZIP environment variable to ``-aL'' and is tempted to add ``-C'' as well. His ZIPINFO variable is set to ``-z''.

DIAGNOSTICS

The exit status (or error level) approximates the exit codes defined by PKWARE and takes on the following values, except under VMS:

0 normal; no errors or warnings detected.

1 one or more warning errors were encountered, but processing completed successfully anyway. This includes zipfiles where one or more files was

skipped due to unsupported compression method or encryption with an unknown password.

2 a generic error in the zipfile format was detected. Processing may have completed successfully anyway; some broken zipfiles created by other archivers have simple work-arounds.

3 a severe error in the zipfile format was detected. Processing probably failed immediately.

4-8 unzip was unable to allocate memory for one or more buffers.

9 the specified zipfiles were not found.

10 invalid options were specified on the command line.

11 no matching files were found.

50 the disk is (or was) full during extraction.

51 the end of the ZIP archive was encountered prematurely.

VMS interprets standard Unix (or PC) return values as other, scarier-looking things, so unzip instead maps them into VMS-style status codes. The current mapping is as follows: 1 (success) for normal exit, 0x7fff0001 for warning errors, and (0x7fff000? + 16*normal_unzip_exit_status) for all other errors, where the `?' is 2 (error) for unzip values 2 and 9-11, and 4 (fatal error) for the remaining ones (3-8, 50, 51). In addition, there is a compilation option to expand upon this behavior: defining RETURN_CODES results in a human-readable explanation of what the error status means.

BUGS

Multi-part archives are not yet supported, except in conjunction with zip. (All parts must be concatenated together in order, and then ``zip -F'' must be performed on the concatenated archive in order to ``fix'' it.) This will definitely be corrected in the next major release.

Archives read from standard input are not yet supported, except with funzip (and then only the first member of the archive can be extracted).

unzip's -M (``more'') option is overly simplistic in its handling of screen output; as noted above, it fails to detect the wrapping of long lines and may thereby cause lines at the top of the screen to be scrolled off before being read. unzip should detect and treat each occurrence of line-wrap as one additional line printed. This requires knowledge of the screen's width as well as its height. In addition, unzip should detect the true screen geometry on all systems.

[MS-DOS] When extracting or testing files from an archive on a defective floppy diskette, if the ``Fail'' option is chosen from DOS's ``Abort, Retry, Fail?'' message, unzip may hang the system, requiring a reboot. Instead, press control-C (or control-Break) to terminate unzip.

Under DEC Ultrix, unzip will sometimes fail on long zipfiles (bad CRC, not always reproducible). This is apparently due either to a hardware bug (cache memory) or an operating system bug (improper handling of page faults?).

Dates and times of stored directories are not restored.

[OS/2] Extended attributes for existing directories are never updated. This is a limitation of the operating system; unzip has no way to determine whether the stored attributes are newer or older than the existing ones.

[VMS] When extracting to another directory, only the [.foo] syntax is accepted for the -d option; the simple Unix foo syntax is silently ignored (as is the less common VMS foo.dir syntax).

[VMS] When the file being extracted already exists, unzip's query only allows skipping, overwriting or renaming; there should additionally be a choice for creating a new version of the file. In fact, the ``overwrite'' choice does create a new version; the old version is not overwritten or deleted.

SEE ALSO

funzip(1L), zip(1L), zipcloak(1L), zipgrep(1L), zipinfo(1L), zipnote(1L), zipsplit(1L)

AUTHORS

The primary Info-ZIP authors (current zip-bugs workgroup) are: Greg ``Cave Newt'' Roelofs (UnZip); Onno van der Linden (Zip); Jean-loup Gailly (compression); Mark Adler (decompression, fUnZip); Christian Spieler (VMS, MS-DOS, shared code, general Zip and UnZip integration); Mike White (Windows GUI, Windows DLLs); Kai Uwe Rommel (OS/2); Paul Kienitz (Amiga, Windows 95); Karl Davis and Sergio Monesi (Acorn RISC OS); George Petrov (MVS, VM/CMS); Harald Denker (Atari, MVS); John Bush (Amiga); Hunter Goatley (VMS); Antoine Verheijen (Macintosh); Chris Herborth (Atari, QNX, BeBox); Johnny Lee (MS-DOS, NT, Windows 95); Steve Salisbury

(NT, Windows 95); and Robert Heath (Windows GUI). The author of the original unzip code upon which Info-ZIP's was based is Samuel H. Smith; Carl Mascott did the first Unix port; and David P. Kirschbaum organized and led Info-ZIP in its early days. The full list of contributors to UnZip has grown quite large; please refer to the CONTRIBS file in the UnZip source distribution for a relatively complete version.

VERSIONS

v1.2	15 Mar 89	Samuel H. Smith
v2.0	9 Sep 89	Samuel H. Smith
v2.x	fall 1989	many Usenet contributors
v3.0	1 May 90	Info-ZIP (DPK, consolidator)
v3.1	15 Aug 90	Info-ZIP (DPK, consolidator)
v4.0	1 Dec 90	Info-ZIP (GRR, maintainer)
v4.1	12 May 91	Info-ZIP
v4.2	20 Mar 92	Info-ZIP (zip-bugs subgroup, GRR)
v5.0	21 Aug 92	Info-ZIP (zip-bugs subgroup, GRR)
v5.01	15 Jan 93	Info-ZIP (zip-bugs subgroup, GRR)
v5.1	7 Feb 94	Info-ZIP (zip-bugs subgroup, GRR)
v5.11	2 Aug 94	Info-ZIP (zip-bugs subgroup, GRR)
v5.12	28 Aug 94	Info-ZIP (zip-bugs subgroup, GRR)
v5.2	30 Apr 96	Info-ZIP (zip-bugs subgroup, GRR)

ProcessZip.exe

ProcessZip.exe is a simpe program for processing of incoming .zip files. It first calls UnZip.exe, and then tries to extract the files spesified on the command line from the archive.

You will typically use this utility to extract file_id.diz or readme.txt files via the Upload Verification module.

If the CRC check is OK, the program will set the exit value to 0. The War FTP Daemon will thereby accept the upload regardless of the extract status.

Sample

The following values in the Upload Verification module will perform a CRC check and first try to extract file_id.diz, and then (if that fails) readme.txt.

Script name: `ProcessZip.exe`
Command line arguments: `$f file_id.diz readme.txt`
Exit status: `Equals to 0`

Note: ProcessZip does not know the path to UnZip.exe. Therefore you must either copy UnZip.exe to a directory withing the scope of your PATH environment variable, or include the War FTP Server's home directory in your PATH environment variable.

If the server reports -3 as return value, this means that UnZip.exe was not found, or could not be started.

The C++ source code for this program is included with the War FTP Daemon distribution to help you write your own scripts.