

File Transfer Protocol

The **File Transfer Protocol** (**FTP**) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

FTP is built on a client-server model architecture using separate control and data connections between the client and the server.^[1] FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is oftensecured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP).

The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems.^{[2][3]} Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into productivity applications, such as web page editors.

Contents

History of FTP servers

Protocol overview

- Communication and data transfer
- Login
- Anonymous FTP
- NAT and firewall traversal
- Differences from HTTP

Web browser support

- Syntax

Security

- FTP over SSH

Derivatives

- FTPS
- SSH File Transfer Protocol
- Trivial File Transfer Protocol
- Simple File Transfer Protocol

FTP commands

FTP reply codes

FTP Servers

See also

References

Further reading

External links

History of FTP servers

The original specification for the File Transfer Protocol was written by Abhay Bhushan and published as RFC 114 on 16 April 1971. Until 1980, FTP ran on NCP, the predecessor of TCP/IP.^[2] The protocol was later replaced by a TCP/IP version, RFC 765 (June 1980) and RFC 959 (October 1985), the current specification. Several proposed standards amend RFC 959, for example RFC 1579 (February 1994) enables Firewall-Friendly FTP (passive mode), RFC 2228 (June 1997) proposes security extensions, RFC 2428 (September 1998) adds support for IPv6 and defines a new type of passive mode.^[4] FTP has two modes: active mode, data transfer, using TCP port 20 and control, command, mode using TCP port 21^[5]

Protocol overview

Communication and data transfer

FTP may run in *active* or *passive* mode, which determines how the data connection is established.^[6] In both cases, the client creates a TCP control connection from a random, usually an unprivileged, port *N* to the FTP server command port 21.

- In active mode, the client starts listening for incoming data connections from the server on port *M*. It sends the FTP command `PORT M` to inform the server on which port it is listening. The server then initiates a data channel to the client from its port 20, the FTP server data port.
- In situations where the client is behind a firewall and unable to accept incoming TCP connections, *passive mode* may be used. In this mode, the client uses the control connection to send a `PASV` command to the server and then receives a server IP address and server port number from the server,^[6] which the client then uses to open a data connection from an arbitrary client port to the server IP address and server port number received!^[7]

Both modes were updated in September 1998 to support IPv6. Further changes were introduced to the passive mode at that time, updating it to *extended passive mode*^[8]

The server responds over the control connection with three-digit status codes in ASCII with an optional text message. For example, "200" (or "200 OK") means that the last command was successful. The numbers represent the code for the response and the optional text represents a human-readable explanation or request (e.g. <Need account for storing file>).^[1] An ongoing transfer of file data over the data connection can be aborted using an interrupt message sent over the control connection.

While transferring data over the network, four data representations can be used.^{[2][3][4]}

- ASCII mode: Used for text. Data is converted, if needed, from the sending host's character representation to 8-bit ASCII before transmission, and (again, if necessary) to the receiving host's character representation. As a consequence, this mode is inappropriate for files that contain data other than plain text.
- Image mode (commonly called Binary mode): The sending machine sends each file byte by byte, and the recipient stores the bytestream as it receives it. (Image mode support has been recommended for all implementations of FTP).
- EBCDIC mode: Used for plain text between hosts using the EBCDIC character set.
- Local mode: Allows two computers with identical setups to send data in a proprietary format without the need to convert it to ASCII.

For text files, different format control and record structure options are provided. These features were designed to facilitate files containing Telnet or ASA.

Data transfer can be done in any of three modes.^{[1][2]}

- Stream mode: Data is sent as a continuous stream, relieving FTP from doing any processing. Rather, all processing is left up to TCP. No End-of-file indicator is needed, unless the data is divided into records.
- Block mode: FTP breaks the data into several blocks (block header, byte count, and data field) and then passes it on to TCP.^[4]
- Compressed mode: Data is compressed using a simple algorithm (usually, run-length encoding).

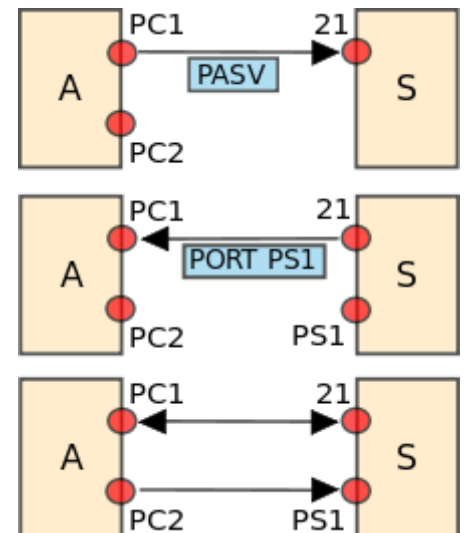


Illustration of starting a passive connection using port 21

Some FTP software also implements a DEFLATE-based compressed mode, sometimes called "Mode Z" after the command that enables it. This mode was described in an Internet Draft, but not standardized.^[9]

Login

FTP login uses normal username and password scheme for granting access.^[2] The username is sent to the server using the USER command, and the password is sent using the PASS command.^[2] This sequence is unencrypted "on the wire", so may be vulnerable to a network sniffing attack.^[10] If the information provided by the client is accepted by the server, the server will send a greeting to the client and the session will commence.^[2] If the server supports it, users may log in without providing login credentials, but the same server may authorize only limited access for such sessions.^[2]

Anonymous FTP

A host that provides an FTP service may provide anonymous FTP access.^[2] Users typically log into the service with an 'anonymous' (lower-case and case-sensitive in some FTP servers) account when prompted for user name. Although users are commonly asked to send their email address instead of a password,^[3] no verification is actually performed on the supplied data.^[11] Many FTP hosts whose purpose is to provide software updates will allow anonymous logins.^[3]

NAT and firewall traversal

FTP normally transfers data by having the server connect back to the client, after the PORT command is sent by the client. This is problematic for both NATs and firewalls, which do not allow connections from the Internet towards internal hosts.^[12] For NATs, an additional complication is that the representation of the IP addresses and port number in the PORT command refer to the internal host's IP address and port, rather than the public IP address and port of the NA

There are two approaches to solve this problem. One is that the FTP client and FTP server use the PASV command, which causes the data connection to be established from the FTP client to the server.^[12] This is widely used by modern FTP clients. Another approach is for the NAT to alter the values of the PORT command, using an application-level gateway for this purpose.^[12]

Differences from HTTP

HTTP essentially fixes the bugs in FTP that made it inconvenient to use for many small ephemeral transfers as are typical in web pages.

FTP has a stateful control connection which maintains a current working directory and other flags, and each transfer requires a secondary connection through which the data are transferred. In "passive" mode this secondary connection is from client to server, whereas in the default "active" mode this connection is from server to client. This apparent role reversal when in active mode, and random port numbers for all transfers, is why firewalls and NAT gateways have such a hard time with FTP. HTTP is stateless and multiplexes control and data over a single connection from client to server on well-known port numbers, which trivially passes through NAT gateways and is simple for firewalls to manage.

Setting up an FTP control connection is quite slow due to the round-trip delays of sending all of the required commands and awaiting responses, so it is customary to bring up a control connection and hold it open for multiple file transfers rather than drop and re-establish the session afresh each time. In contrast, HTTP originally dropped the connection after each transfer because doing so was so cheap. While HTTP has subsequently gained the ability to reuse the TCP connection for multiple transfers, the conceptual model is still of independent requests rather than a session.

When FTP is transferring over the data connection, the control connection is idle. If the transfer takes too long, the firewall or NAT may decide that the control connection is dead and stop tracking it, effectively breaking the connection and confusing the download. The single HTTP connection is only idle between requests and it is normal and expected for such connections to be dropped after a time-out.

Web browser support

Most common web browsers can retrieve files hosted on FTP servers, although they may not support protocol extensions such as FTPS.^{[3][13]} When an FTP—rather than an HTTP—URL is supplied, the accessible contents on the remote server are presented in a manner that is similar to that used for other web content. A full-featured FTP client can be run within Firefox in the form of an extension called FireFTP.

Syntax

FTP URL syntax is described in RFC 1738, taking the form: `ftp://[user[:password]@]host[:port]/url-path` (the bracketed parts are optional).

For example, the URL `ftp://public.ftp-servers.example.com/mydirectory/myfile.txt` represents the file *myfile.txt* from the directory *mydirectory* on the server *public.ftp-servers.example.com* as an FTP resource. The URL `ftp://user001:secretpassword@private.ftp-servers.example.com/mydirectory/myfile.txt` adds a specification of the username and password that must be used to access this resource.

More details on specifying a username and password may be found in the browsers' documentation (e.g., Firefox^[14] and Internet Explorer^[15]). By default, most web browsers use passive (PSV) mode, which more easily traverses end-user firewalls.

Some variation has existed in how different browsers treat path resolution in cases where there is a non-root home directory for a user.^[16]

Security

FTP was not designed to be a secure protocol, and has many security weaknesses.^[17] In May 1999, the authors of RFC 2577 listed a vulnerability to the following problems:

- Brute force attack
- FTP bounce attack
- Packet capture
- Port stealing (guessing the next open port and usurping a legitimate connection)
- Spoofing attack
- Username enumeration

FTP does not encrypt its traffic; all transmissions are in clear text, and usernames, passwords, commands and data can be read by anyone able to perform packet capture (sniffing) on the network.^{[2][17]} This problem is common to many of the Internet Protocol specifications (such as SMTP, Telnet, POP and IMAP) that were designed prior to the creation of encryption mechanisms such as TLS or SSL.^[4]

Common solutions to this problem include:

1. Using the secure versions of the insecure protocols, e.g. FTPS instead of FTP and TelnetS instead of Telnet.
2. Using a different, more secure protocol that can handle the job, e.g. SSH File Transfer Protocol or Secure Copy Protocol.
3. Using a secure tunnel such as Secure Shell (SSH) or virtual private network (VPN).

FTP over SSH

FTP over SSH is the practice of tunneling a normal FTP session over a Secure Shell connection.^[17] Because FTP uses multiple TCP connections (unusual for a TCP/IP protocol that is still in use), it is particularly difficult to tunnel over SSH. With many SSH clients, attempting to set up a tunnel for the control channel (the initial client-to-server connection on port 21) will protect only that channel; when data is transferred, the FTP software at either end sets up new TCP connections (data channels) and thus have nonconfidentiality or integrity protection.

Otherwise, it is necessary for the SSH client software to have specific knowledge of the FTP protocol, to monitor and rewrite FTP control channel messages and autonomously open new packet forwardings for FTP data channels. Software packages that support this mode include:

- Tectia ConnectSecure (Win/Linux/Unix)^[18] of SSH Communications Security's software suite

Derivatives

FTPS

Explicit FTPS is an extension to the FTP standard that allows clients to request FTP sessions to be encrypted. This is done by sending the "AUTH TLS" command. The server has the option of allowing or denying connections that do not request TLS. This protocol extension is defined in RFC 4217. Implicit FTPS is an outdated standard for FTP that required the use of a SSL or TLS connection. It was specified to use different ports than plain FTP

SSH File Transfer Protocol

The SSH file transfer protocol (chronologically the second of the two protocols abbreviated SFTP) transfers files and has a similar command set for users, but uses the Secure Shell protocol (SSH) to transfer files. Unlike FTP, it encrypts both commands and data, preventing passwords and sensitive information from being transmitted openly over the network. It cannot interoperate with FTP software.

Trivial File Transfer Protocol

Trivial File Transfer Protocol (TFTP) is a simple, lock-step FTP that allows a client to get a file from or put a file onto a remote host. One of its primary uses is in the early stages of booting from a local area network, because TFTP is very simple to implement. TFTP lacks security and most of the advanced features offered by more robust file transfer protocols such as File Transfer Protocol. TFTP was first standardized in 1981 and the current specification for the protocol can be found in RFC 1350.

Simple File Transfer Protocol

Simple File Transfer Protocol (the first protocol abbreviated SFTP), as defined by RFC 913, was proposed as an (unsecured) file transfer protocol with a level of complexity intermediate between TFTP and FTP. It was never widely accepted on the Internet, and is now assigned Historic status by the ETF. It runs through port 115, and often receives the initialism of *SFTP*. It has a command set of 11 commands and support three types of data transmission: ASCII, binary and continuous. For systems with a word size that is a multiple of 8 bits, the implementation of binary and continuous is the same. The protocol also supports login with user ID and password, hierarchical folders and file management (including *rename*, *delete*, *upload*, *download*, *download with overwrite*, and *download with append*).

FTP commands

FTP reply codes

Below is a summary of FTP reply codes that may be returned by an FTP server. These codes have been standardized in RFC 959 by the IETF. The reply code is a three-digit value. The first digit is used to indicate one of three possible outcomes — success, failure, or to indicate an error or incomplete reply:

- 2yz – Success reply
- 4yz or 5yz – Failure reply
- 1yz or 3yz – Error or Incomplete reply

The second digit defines the kind of error:

- x0z – Syntax. These replies refer to syntax errors.
- x1z – Information. Replies to requests for information.
- x2z – Connections. Replies referring to the control and data connections.
- x3z – Authentication and accounting. Replies for the login process and accounting procedures.
- x4z – Not defined.
- x5z – File system. These replies relay status codes from the server file system.

The third digit of the reply code is used to provide additional detail for each of the categories defined by the second digit.

FTP Servers

Some popular open source FTP server implementations are:

- [FileZilla Server](#) (Windows)
- [Pure-FTPd](#) (Unix)
- [Vsftpd](#) (Unix)
- [ProFTPd](#) (Unix)
- [CrushFTP](#) (Mac, Win, Linux)
- [Rumpus](#) (Mac)
- [WingFTP](#) (Mac, Win)

See also

- | | |
|--|---|
| ▪ Comparison of FTP client software | ▪ FTPFS |
| ▪ Comparison of FTP server software | ▪ List of FTP commands |
| ▪ Comparison of file transfer protocols | ▪ List of FTP server return codes |
| ▪ Curl-loader – FTP/S loading/testing open-source software | ▪ List of FTP server software |
| ▪ File eXchange Protocol(FXP) | ▪ Managed File Transfer |
| ▪ File Service Protocol(FSP) | ▪ OBEX |
| ▪ FTAM | ▪ Shared file access |
| | ▪ TCP Wrapper |

References


1. Forouzan, B.A. (2000). *TCP/IP: Protocol Suite* (1st ed.). New Delhi, India: Tata McGraw-Hill Publishing Company Limited.
2. Kozierok, Charles M. (2005). "The TCP/IP Guide v3.0" (http://www.tcpipguide.com/free/t_FTPOverviewHistoryandStandards.htm). Tcpiipguide.com.
3. Dean, Tamara (2010). *Network+ Guide to Networks* Delmar. pp. 168–171.
4. Clark, M.P. (2003). *Data Networks IP and the Internet* (1st ed.). West Sussex, England: John Wiley & Sons Ltd.
5. List of TCP and UDP port numbers
6. "Active FTP vs. Passive FTP, a Definitive Explanation" (<https://web.archive.org/web/20110504071617/http://slacksite.com/other/ftp.html>) Slacksite.com. Archived from the original (<http://slacksite.com/other/ftp.html>) on 4 May 2011.
7. [RFC 959](https://tools.ietf.org/html/rfc959) (<https://tools.ietf.org/html/rfc959>) (Standard) File Transfer Protocol (FTP). Postel, J. & Reynolds, J. (October 1985).
8. [RFC 2428](https://tools.ietf.org/html/rfc2428) (<https://tools.ietf.org/html/rfc2428>) (Proposed Standard) Extensions for IPv6, NAT, and Extended Passive Mode. Allman, M. & Metz, C. & Ostermann, S. (September 1998).
9. Preston, J. (January 2005). *Deflate transmission mode for FTP* (<https://tools.ietf.org/html/draft-preston-ftpext-deflate-03.txt>). IETF. I-D draft-preston-ftpext-deflate-03.txt <https://tools.ietf.org/html/draft-preston-ftpext-deflate-03.txt> Retrieved 27 January 2016

10. Prince, Brian. "Should Organizations Retire FTP for Security?"(<http://www.securityweek.com/should-organizations-retire-ftp-security>) *Security Week*. Security Week. Retrieved 14 September 2017.
11. RFC 1635 (<https://tools.ietf.org/html/rfc1635>)(Informational) How to Use Anonymous FTP. & Emtage, A. & Marine, A. (May 1994).
12. Gleason, Mike (2005). "The File Transfer Protocol and Your Firewall/NAT" (http://www.ncftp.com/ncftpd/doc/misc/ftp_and_firewalls.html) Ncftp.com.
13. Matthews, J. (2005). *Computer Networking: Internet Protocols in Action*(1st ed.). Danvers, MA: John Wiley & Sons Inc.
14. "Accessing FTP servers | How to | Firefox Help"(http://support.mozilla.com/en-US/kb/Accessing+FTP+servers#FTP_servers_that_require_a_username_and_password)Support.mozilla.com. 2012-09-05 Retrieved 2013-01-16.
15. "How to Enter FTP Site Password in Internet Explorer"(<http://support.microsoft.com/kb/135975>) Support.microsoft.com. 2011-09-23 Retrieved 2015-03-28. Written for IE versions 6 and earlier Might work with newer versions.
16. Jukka "Yucca" Korpela (1997-09-18). "FTP URLs" (<https://www.cs.tut.fi/~jkorpela/ftputl.html>) "IT and communication" (www.cs.tut.fi/~jkorpela/) Retrieved 2016-01-06.
17. "Securing FTP using SSH"(<http://www.nurdletech.com/linux-notes/ftp/ssh.html>). Nurdletech.com.
18. "Access using SSH keys & PCI DSS compliance"(<http://ssh.com/index.php/products/tectia-pci-point-to-point-encryption.html>). *ssh.com*.

Further reading

- RFC 697 – CWD Command of FTP July 1975.
- RFC 959 – (Standard) File Transfer Protocol (FTP). J.Postel, J. Reynolds. October 1985.
- RFC 1579 – (Informational) Firewall-Friendly FTP February 1994.
- RFC 1635 – (Informational) How to Use Anonymous FTP May 1994.
- RFC 1639 – FTP Operation Over Big Address Records (FOOBAR). June 1994.
- RFC 1738 – Uniform Resource Locators (URL). December 1994.
- RFC 2228 – (Proposed Standard) FTP Security Extensions. October 1997.
- RFC 2389 – (Proposed Standard) Feature negotiation mechanism for the File Transfer Protocol. August 1998.
- RFC 2428 – (Proposed Standard) Extensions for IPv6, NAT, and Extended passive mode. September 1998.
- RFC 2577 – (Informational) FTP Security Considerations. May 1999.
- RFC 2640 – (Proposed Standard) Internationalization of the File Transfer Protocol. July 1999.
- RFC 3659 – (Proposed Standard) Extensions to FTP. Hethmon. March 2007.
- RFC 5797 – (Proposed Standard) FTP Command and Extension Registry March 2010.
- RFC 7151 – (Proposed Standard) File Transfer Protocol HOST Command for Virtual Hosts. March 2014.
- IANA FTP Commands and Extensions registry- The official registry of FTP Commands and Extensions

External links

-  [Communication Networks/File Transfer Protocol](#) at Wikibooks
 - [FTP Server Online Tester](#) Authentication, encryption, mode and connectivity
-

Retrieved from 'https://en.wikipedia.org/w/index.php?title=File_Transfer_Protocol&oldid=859469006

This page was last edited on 14 September 2018, at 07:21(UTC).

Text is available under the [Creative Commons Attribution-ShareAlike License](#)additional terms may apply By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.