# FTP

In-script FTP client

```
const ftp = require('ftp')
```

The File Transfer Protocol (FTP) module allows scripts to access an FTP server.

**Note:** You must have access to an FTP server and know the following values: **host**, **port**, **username**, and **password**.

**Tip**: To enable FTP for your environment, please contact support.

## module FTP

### create(options)

Connects an FTP client instance to a remote FTP server.

**Tip:** If you set the `host` with an IP address, you should also set the secureOption `rejectUnauthorized` to `false` to avoid potential conflicts with the domain name on the server certificate, as provided by the TLS Server Name Indication (SNI) extension.

*Arguments*

- `options` { Object }
  - `host` { String = *localhost* } FTP server domain name or IP address.
  - `port` { Number = `21` } FTP server port.
  - `username` { String = `anonymous` } Username for authentication.
  - `password` { String = `anonymous@` } Password for authentication.
  - `connTimeout` { Number = `10000` } Wait time in milliseconds to establish a control connection.

- `pasvTimeout` { Number = `10000` } Wait time in milliseconds to establish a `PASV` data connection.
- `keepalive` { Number = `10000` } Wait time in milliseconds to send a dummy (NOOP) command to keep the connection alive.
- `secure` { String = `true` } Always set to `true` for control and data encryption.
- `secureOptions` { Object } Additional options to be passed to `tls.connect()`, as listed below. For more details, view the Node.js TLS module reference.
  - `ca` { String, String[], Buffer, Buffer[] } Overrides default Certificate Authority (CA) certificates.
  - `cert` { String, String[], Buffer, Buffer[] } Defines certification chains in Privacy-Enhanced Mail (PEM) format.
  - `sigalgs` { String } Defines a list of supported signature algorithms.
  - `ciphers` { String } Replaces the cipher suite specification.
  - `crl` { String, String[], Buffer, Buffer[] } Defines PEM-formatted Certificate Revocation Lists (CRLs).
  - `dhparam` { String, Buffer } Defines Diffie–Hellman parameters.
  - `ecdhCurve` { String } Describes a named curve or a colon-separated list of curve NIDs or names to use for Elliptic-curve Diffie–Hellman (ECDH) key agreement.
    **Default:** `tls.DEFAULT_ECDH_CURVE`.
  - `honorCipherOrder` { Boolean } Attempts to use the cipher suite preferences of the server instead of the client.
  - `key` { String, String[], Buffer, Buffer[], Object[] } Sets private keys in PEM format.
  - `minVersion` { String } Allows you to choose the minimum allowable TLS version from the following: `'TLSv1.3'`, `'TLSv1.2'`, `'TLSv1.1'`, or `'TLSv1'`.
    **Default:** `tls.DEFAULT_MIN_VERSION`.
  - `maxVersion` { String } Allows you to choose the maximum allowable TLS version from the following: `'TLSv1.3'`, `'TLSv1.2'`, `'TLSv1.1'`, or `'TLSv1'`. This option cannot be used in conjunction with `secureProtocol`; you can only use one or the other.
    **Default:** `tls.DEFAULT_MAX_VERSION`.
  - `passphrase` { String } Shared passphrase used for a single private key or a PFX.
  - `pfx` { String, String[], Buffer, Buffer[], Object[] } Defines the PFX or PKCS12 encoded private key and certificate chain.
  - `secureProtocol` { String } Provides a legacy mechanism to select the TLS protocol version to use. It does not support independent control of the minimum and maximum versions. It also does not support

    limiting the protocol to TLSv1.3. If you wish to do so, use `minVersion` and `maxVersion` instead. **Default:** none, see `minVersion`.
  - `rejectUnauthorized` { Boolean = `true` } If `true`, the server certificate is verified against the supplied list of CAs.

*Returns*

- { ftp.Client } A new client instance.

### list()

Lists current script connections.

*Returns*

- { Client[] } Active connections for this script.

## class ftp.Client

To create an FTP client, use ftp.create(options).

### chmod(path, mode)

Changes file permissions.

*Arguments*

- `path` { String } The resource to modify.
- `mode` { String } A string containing octal numbers.

### close()

Closes the connection with the server.

**Note:** Connections are automatically closed when a script exits, which frees up resources to open another connection.

## delete(path)

Deletes a file on the server.

*Arguments*

- `path` { String } File path.

## get(path)

Downloads data from the server.

*Arguments*

- `path` { String } File path.

*Returns*

- { Buffer } A buffer containing the file data.

## list(path)

Lists the contents of a remote directory.

*Arguments*

- `path` { String } The path to the directory. The default is the current working directory.

*Returns*

- { Object[] } A list of directory contents.
  - `type` { String } Entry type:
    - `d`: directory
    - `-`: file
    - `l`: symlink
  - `name` { String } Entry name.

- `size` { Number } Entry size in bytes.
- `modifyTime` { date } Last modified date of the entry.
- `rights` { Object } Entry access rights.
    - `username` { String } User access rights (rwx).
    - `group` { String } Group access rights (rwx).
    - `other` { String } Other access rights (rwx).
- `owner` { String } Owner name or ID.
- `group` { String } Group name or ID.
- `target` { String } Symlink target.
- `sticky` { Boolean } `True` if sticky bit is set.

## mkdir(path)

Creates a new directory on the server.

*Arguments*

- `path` { String } The directory to create.
- `recursive` { Boolean = *false* } Creates intermediate directories as required.

## put(path, data)

Uploads data to the server.

*Arguments*

- `path` { String } The file name.
- `data` { Buffer } A buffer containing the file data.

## rename(oldPath, newPath)

Renames a file on the server.

*Arguments*

- `oldPath` { String } Old file name.
- `newPath` { String } New file name.

## Example

```
const ftp = require('ftp'),

   conn = ftp.create({
      host: '<hostname>',
      port: '<port>',
      username: '<username>',
      password: '<password>',
      secureOptions: {
          rejectUnauthorized: false,
           ciphers: 'TLS_AES_256_GCM_SHA384',
           minVersion: 'TLSv1.3'
       }
   }),
   buffer = new Buffer('ftp works!')

conn.mkdir('/one/two', true)
conn.put('one/two/test.txt', buffer)
let list = conn.list('/one/two')

conn.close()

return list
conn.close
```

---

Source: [FTP - Medable]

Accessed on 2022-07-21