# Language/Runtime environments/Modules used

* Language: Typescript
* Runtime environment: Nodejs
* This code uses node.js native modules ‘net’ and ‘readline’

# xFx Protocol definition

* The server opens a connection passively on localhost IP on port 59090
* The client then opens an active connection to the server at localhost:59090 and gets assigned a random port
* I defined the protocol to always have an event and data related to that specific event
* When the client establishes connection to the server the client process will list all the available commands that can be executed
* If the user chooses to list the available files to download
  + The client process will send a buffer containing an object with the event ‘list’
  + The server process will respond with an object containing an event ‘list:response’ and the available files concatenated as data
  + The client process will receive the ‘list:response’ object and display the data for the user
  + The client terminates the connection.
* If the client chooses to download one of the files available in the server machine
  + The client process will send an object containing the event ‘download:request’ and will as data: the name of the file, and the size of the file if it already exists on the client machine, otherwise it will only send the name of the file.
  + If the server process doesn’t receive a file size It will send the whole file as base64 encoding as the event ‘download: file’ that the client process will receive, decode, and create a file with that file name.
  + If the server process receives a file size with the ‘download:request’ event, it will get the same size from the original file and send a md5 signature to the client process as the ‘download:signature’ event.
  + The client process will then also has the existing file, if it is a match it will send a ‘download:start’ event with the following data: the file name and the size to start downloading from. If the signatures are a no match, the client will delete the existing file as it became a dirty/stale copy and send a ‘download:request’ with no size so that the server process sends the file from the first byte.
  + When the server process receives a ‘download:start’ event from the client process, it knows that signatures have been verified and it will slice the file starting from the already existing and verified size and will send out the remaining part as a base64 string.
  + The client decode base64 string and write the file.
  + The client then terminates connection.
* If the client chooses to upload a file to the server machine
  + The client process will send an object with the event ‘upload:request’ and the name of the file as data.
  + The server process will then receive the object and check if there exists a file with that specific name or not. If it does, the server will send an event ‘upload:signature’ with the following data: the file name, the size already uploaded and the md5 signature of that uploaded size. If there is no file with that name, the server will send no signature and a size of 0
  + If the client receives a size of 0, it will send an object with the event ‘upload:file’ and the file buffer converted to a base64 string. If the size of the already uploaded file is not 0, The server process will compare the hash of the already uploaded file with the hash of the original file to upload sliced from the first byte to the size sent with the request
  + If the hashes match, the client process will send an object with the event ‘upload:file’ with the following data: the file name, OK meaning there is a checksum match, and the remaining bytes to upload. If the hashes don’t match, the server will send an object with the same event and file name but will send a N meaning the hashes don’t match and the file buffer from the start as a base64 string
  + If the client receives an OK, it will append the decoded base64 buffer to the already uploaded size, otherwise, it will delete the already uploaded chunks of the file as they have become dirty or stale and will write the file from the start.
  + The client will then terminate connection.