

File Protocol Specfication for Calc Projects

Header

This is the header that is sent over the network. Note the xor_byte field specifies the byte to xor the xor_key with. Example, xor_byte = 0x55 xor_key 0x11223344: $\text{key_to_xor_payload} = 0x11223344 \wedge 0x55555555$. Every payload is xor encrypted with that value.

member	type	xor_byte	reserved	payload_len	xor_key
Length (Bytes)	1	1	2	4	4

Types

These are the currently supported types of messages that fit into the type field. Note these describe the payload type and have corressponding lengths with each type. If the length is variable, please see the payload_len field.

Type	Length	Purpose
FILE_INFO	1	Contains information about the client or server. Hostname, server up-time, multi-thread support, encryption support) See file_info enum.
FILE_METADATA	Variable	Contains file id's as a list, each id separated by 4 NULL bytes or a one-byte meta code
FILE_DATA	Variable	Contains file data for a single equ file
FILE_SHUTDOWN	0	Signal to shutdown if operating in server mode. Report FILE_ERROR if received in client mode
FILE_FORWARD	Variable	Contains list of IP address or unix domain sockets to forward all solutions to.
FILE_WAIT	4	Contains a single unsigned integer specifying number of second to wait before returning the solutions to the server/client
FILE_RESET	0	Signal to reset communications. Respond with FILE_INFO query/response
FILE_ERROR	8	Signal an error in solutions or file. Include file id
FILE_SUCCESS	8	Signal correct solutions or file. Include file id

Protocol

There are 2 modes to operate in client/server. The client will beacon out and the server will wait for a trigger.

Client

When operating in client mode, the client will beacon out to a predefined ip address or unix domain socket file, given as a command line argument. The client should follow the protocol sequence:

1. Beacon with `FILE_INFO`. If the server is compatible with the given information, it will respond with a `FILE_INFO` response, which contains the information the server is capable of handling. If the server is not compatible, a `FILE_RESET` response is sent from the server and the client should wait 1 minute before beaoning again. If the server determines an out of sync message was received, it will send a `FILE_SHUTDOWN`.
2. Once a compatibility match has been reached, the server will start to send `FILE_DATA` packets, each containing a single file. The data will be encrypted if encryption is supported. Upon receiving the file, send a `FILE_METADATA` message with `META_RECV` and parse the contents. If any errors occur, respond with `FILE_ERROR`, and discard the file. Once the file is parsed, save the file and wait for either another `FILE_DATA` or a `FILE_FORWARD` message which will specify where to send the solutions. The client should be prepared to handle a `FILE_WAIT` or `FILE_RESET` query. The last command will always be a `FILE_METADATA` query. If the `FILE_METADATA` query has a list of `file_ids`, solve the ids listed and cleanup the rest of the files.
3. Upon completion of solving the files, if a `FILE_FORWARD` command was received and an address to send the solutions has been added, the client must ensure the new destination is compatible and then send a `FILE_METADATA` query saying that solutions are ready to all destinations. Send files to only the compatible destinations. If the address did not change, just send the `FILE_METADATA` query with solutions are ready. Send the solutions files with a `FILE_DATA` response, and the server will respond with `FILE_SUCCESS` or `FILE_ERROR`. In either case, continue to send the solution files. Once all solution files have been sent and responses from the server are received all correlated by `file_id`, send a `FILE_SHUTDOWN` and close the connection.

Server

When operating in server mode, the server will wait for a call-in and only accept those clients that are compatible with the server's configuration.

1. The server should expect a `FILE_INFO` query before any other command and respond with it's current running configuration. If any other command is received, send a `FILE_SHUTDOWN` to close the connection and keep track of command errors. If more than three errors are received, block all connections from that address.
2. Next, the server will send a `FILE_DATA` command with the file to solve. If `FILE_METADATA` with `META_RECV` is received, continue sending files. If any errors (`FILE_ERROR`) are reported in parsing, note the error and discard the file. If the private configuration states the files should be forwarded, send a `FILE_FORWARD` command at the end of sending all of the files. If the private configuration states the client should wait before beaoning with solutions, send a `FILE_WAIT` command at the end of sending all of the files. Next, send a `FILE_METADATA` query with `META_SOLVE` sub command. After, the server should wait for the solutions to be solved.
3. When the client is ready to deliver solutions, ensure a `FILE_METADATA` query with `META_READY` sub command is received. Then receive and check all solutions against the

files. Respond with `FILE_SUCCESS` if correct, and `FILE_ERROR` if not correct. Once all solutions received, expect a `FILE_SHUTDOWN` command. If `FILE_SHUTDOWN` is not received within 5 seconds, send a `FILE_RESET` command.