IotaGFTapp Commands

The IotaGFTapp listens at the TCP port 127.0.0.1:33001 when running.

Any program can connect to that port and send command strings to setup up a recording session.

Here is an example Python script (setUTCeventTime.py) to fill the UTC event data/time entry box:

```
import socket, os
def sendUTCeventStartTime():
    HOST = '127.0.0.1'
    PORT = 33001
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        # Make the command string MSGLEN in length
        msg = makeMsg('setUTCeventTime 2026-01-25 10:11:12')
            s.sendall(msg)                          # Send the command
            _, ans = getResponse(s)                 # Get the response
            ans = msgTrim(ans.decode("utf-8"))      # Turn bytes into string characters
            print('Sent:', 'setUTCeventTime 2026-01-25 10:11:12')
            if ans == "OK":
                print("Rcvd:", ans)
            else:
                print("ERR!:", ans)
sendUTCeventStartTime()
```

In this example, the UTC time is hard coded, and the script will need to be modified to accept a ‘real’ event time from some source. It would be nice if this type of script could be added to Occult Watcher (OW) in such a way that the user can ask OW to send an event time to IotaGFTapp.

makeMsg() is used to pad strings so that they contain a fixed number of characters (1000). This is done to avoid any possibility of a command being delivered in more than

one part, which the TCP protocol allows – IotaGFTapp always waits for a complete 1000 character message.

msgTrim() removes the trailing spaces that were added to make a message of length MSGLEN.

makeMsg() and msgTrim() are defined in the startup script SharpCapServer.py

All commands will receive a response string. If the command was valid and could be executed, the response will always be OK If errors are possible, they are shown below each command in red.

Here are the command strings that IotaGFTapp will recognize and act on when sent to 127.0.0.1:33001:

1) setAutorunTrue

2) setAutorunFalse

3) setShutdownTrue

4) setShutdownFalse

5) setUTCeventTime yyyy-mm-dd hh:mm:ss
   Invalid UTC time format

6) setUTCeventTime

7) recordingTime xx
   Invalid recording time

8) armUTCstart
   No camera selected
   Invalid recording time
   Invalid UTC date/time
   Start time is in the past by xx seconds

9) flash now

10) flash duration xx
    Invalid flash duration

11) setLEDintensity xxx
    Invalid intensity value

12) setLEDon

13) setLEDoff

If a string that is not in the above list is sent, Unimplemented command will be returned as an error message.

Commands 1 and 2 set the desired checked-state of the auto-run FitsReader checkbox in the GUI.

Commands 3 and 4 set the state of the shutdown CPU at end-of-recording checkbox in the GUI.

Command 5 sets the text in the UTC event date/time entry box. Note: the format shown is strict and must be adhered to.

Command 6 must contain only the characters shown – no trailing spaces allowed. This clears the UTC event date/time entry box and sets up a special test where a recording will be made starting 10 seconds after the schedule is armed. This facilitates testing whether the camera is properly configured for the observation recording.

Command 7 sets the desired length of the recording between the flash goalposts. The ‘event’ will be centered in this part of the recording. The units are seconds.

Command 8 ‘arms’ the recording schedule.

Command 10 requires a positive duration in seconds.

Command 11 requires an intensity value of 0 to 765.