Legacy TCS Network Communication

Overview

Legacy TCS only allows remote telescope communication and control through a serial port on the main TCS computer. In order to allow network communication and control of the telescope we have built software called Telcom to convert tcp/ip sockets to serial communication. To make sure this software is up and running on the BOK telescope you can check the website http://bokpop.as.arizona.edu and look for the table marked telemetry. This webpage queries the telcom server about once a second.

Request syntax

The Telcom server makes 14 pieces of information available.

Description	Key Word	return synax
Motion Status	MOT	0=No motion 1=RA Motion 2=Dec Motion 3=RA and Dec Motion
Right Ascension	RA	[hhmmss.ss]
Declination	DEC	[+/-ddmmss.ss]
Hour Angle	HA	[+/-hh:mm:ss.ss]
Secant z (Air Mass)	SECZ	[s.ss]
Epoch	EQ	D[YYYY.YYY]
Julian Date	JD	[DDDDDDD.D]
Universal Time	UT	[hh:mm:ss.s]
IIS position	IIS	[DDD.D] angle in degrees
Local Sidereal Time	ST	[hh:mm:ss.ss]
Elevation Angle	EL	[DD.D]
Azimuth Angle	AZ	[DDD.D]
Dome positoin	DOME	[DDD.D] this position is relative to the telescope azimuth. To get absolute dome position add this angle to the telescope azimuth.
WOBBLE	WOBBLE	???

To retrieve these values simply open a TCP socket to the Telcom server and send a request. The syntax is as follows.

<TELESCOPE ID> TCS <REF NUM> REQUEST <KEYWORD>

For Example:

BOK TCS 123 REQUEST RA

The telcom server will return:

<TELESCOPE ID> TCS <REF NUM> <RETURN VALUE>

For Example:

BOK TCS 123 214412.79

As you can see the RA units are not seperated by colons.

"ALL" keyword

If you want to get multiple pieces of data from the telcom server rapidly you can use the ALL keyword. This will return a string containing all the telemetry data described above which will have to be parsed. The string gives values seperated by 1 or more white spaces. The table below shows the order the values are placed in the string.

Return Values	MOT	RA	DEC	HA	ST	EL	AZ	SECZ	EQ	JD	WOBBLE	DOME	UT	IIS
Example Value	0	221941.92	+314308.8	-00:00:00	22:20:19	90.0	+7.6	1.00	D2000.000	2456698.3	1	-78.6	20:27:19.1	0.0

An example string:

```
BOK TCS 123 0 2231:27.64 +314312.0 +00:00:01 22:32:22 90.0 -29.2 1.00 2000.000 2456702.3 1 -115.4 20:23:36.5 0.0
```

It should be noted that all strings returned on the Telcom socket will end with a carriage return (\r).

Command Syntax

The syntax for sending commands to the Telcom server is very similar to sending requests. The syntax looks like:

<TELESCOPE ID> TCS <REF NUM> <COMMAND> <ARG_1> … <ARG_n>

For Example

BOK TCS 123 BIASRA 12.5

This example sets the Right Ascension bias rate at 12.5 seconds per second.

Each command will return:

<TELESCOPE ID> TCS <REF NUM> <ERROR_STATE>

For Example

BOK TCS 123 OK

The <ERROR_STATE> value will be either “OK” or “BAD”. “OK” means the command was understood and “BAD” means the command was NOT understood (not found in the list of commands). “OK” does NOT mean the command was executed. For example, if the Right Ascension drive is turned off or disable and you send the command:

BOK TCS 123 TRACKON

Telcom server will return :

BOK TCS 123 OK

But the telescope will not start tracking because it can't when the drives are off.

***For brevity sake the complete list of TCS commands will not be written here. Instead they can be found list of network commands

Example Scripts

Below are some sample scripts to communicate with Telcom server

telem_example.py

This program gets the telemetry from The Telcom server and stores it in a python dictionary object and prints it out in a clean format.

```
#!/usr/bin/python

"""
Author: Scott Swindell, Steward Observatory

A short example script to grab telemetry from the BOK Telcom server
the Telcom server and store that information in a python
dictionary object"""

IPADDR = ""     #Available upon request
PORT  = ""      #Available upon request

TELID = "BOK"
REF_NUM = 123
REQUEST = "ALL"

keyList = [
        "ID"            ,
        "SYS"           ,
        "REF_NUM"       ,
        "MOT"           ,
        "RA"            ,
        "DEC"           ,
        "HA"            ,
        "ST"            ,
        "EL"            ,
        "AZ"            ,
        "SECZ"          ,
```

```
        "EQ"            ,
        "JD"            ,
        "WOBBLE"        ,
        "DOME"          ,
        "UT"            ,
        "IIS"



                ]

import socket

#instantiate the socket class
telSock = socket.socket( socket.AF_INET, socket.SOCK_STREAM )
telSock.settimeout( 0.5 )

#IPADDR = telSock.gethostbyname(HOST)
telSock.connect( ( IPADDR, PORT ) )#open the socket

reqString = "%s TCS %i REQUEST %s" %( TELID, REF_NUM, REQUEST )


telSock.send( reqString ) #send the request

resp = ""
test = True

#Grab 100 bytes at a time
#from the socket and check for
#timeouts.
while test:
        try:
                inStuff = telSock.recv( 100 )

        except socket.timeout:
                if resp:
                        print "No Telem Recieved"
                test = False
                break

        if inStuff:
                resp+=inStuff

        else:
                test=False

#turn string into list, seperate by whitespace
resp = resp.split(' ')
```

```
cleanResp = []

#remove all empty elements and new line
#and put everything else in cleanResp
for char in resp:
        if char != '' and not char.endswith("\n"):
                cleanResp.append( char )


#gather the telemetry into a dictionary
#for easy referencing
telemDict = {}
II = 0
for key in keyList:
        telemDict[key] = cleanResp[II]
        II+=1

#print them all out in
#a nice clean way
for (key, value) in telemDict.iteritems():
        print "|%s\t|\t%s|" %( key.ljust( 10 ), value.ljust( 12 ) )
```