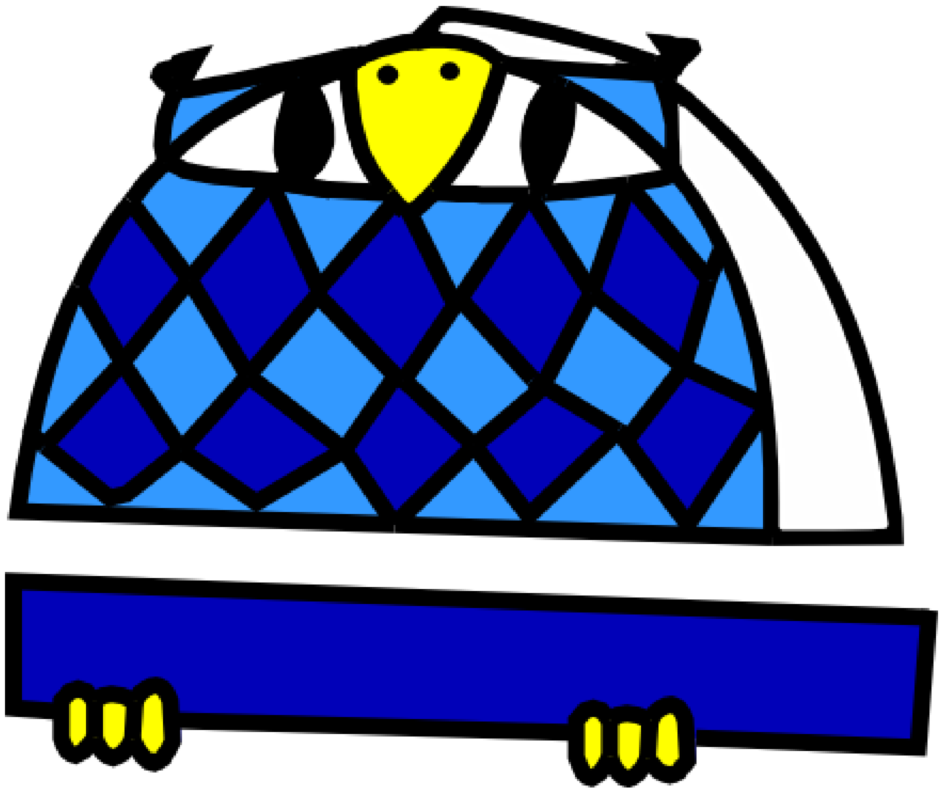
GNS
User's manual



By Lunático Astronomía

INDEX

1. Preface

Thank you for your interest in the **Good Night System**, our innovative product to make your long imaging sessions easier.

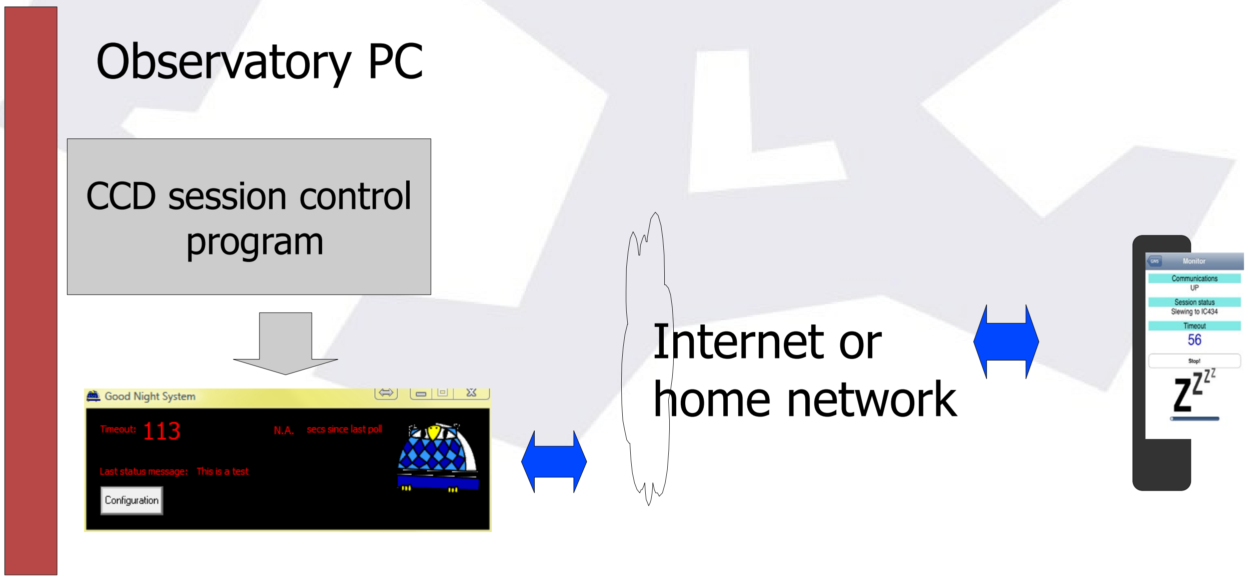
Features that set apart the **Good Night System**:

- **it does not rely on SMSs or emails**; to the contrary, the smartphone constantly plays an active role.
- **safety goes first**: in some cases you may be warned when no real danger exists, most probably if the time allotted for a task is exceeded (because it was too short) or the communications go down; but provided the smartphone is powered, it will wake you up if something goes wrong or *seems to go wrong.*
- both the Windows application and the smartphone app are **small and reliable** by design, and have been **extensively tested**, for a period of several months by a group of real users, under real circumstances.

As you already know, the "**Good Night System**" comprises two elements, both needed to ensure observatory protection:

a) a small windows program
b) a smartphone app

It will work as shown in the following drawing:



So at one side, we'll have our PC GNS program being updated by our ccd/session control software of choice. At the other side, our smartphone (any number of smartphones, actually) will be asking the PC about the progress of the session.

In order to have it up and running, there are a few tasks to perform, to make sure everything will run fine when in real conditions. We have to get the smartphone “talking” to the PC, and then we have to get our automation program updating the PC program.

Let's start from the PC side.

2. Installing and checking the PC program

Brief foreword about 32 and 64 bits systems: the GNS software will run in both systems, and is compatible with both 32 and 64 bits software.

- **Install the software[1]:** it can be downloaded from the **GNS** page of our web site (https://lunaticoastro.com) – the software should install automatically under any "current" Windows version (from XP to Windows 10). The only requisite is to have the .NET platform already installed, which is most likely the case if you are into observatory automation as it is also required by many programs and by the ASCOM platform.

  In case you don't, please visit Microsoft's .NET download page:
  http://www.microsoft.com/net/download
  … and select the platform suitable for your system. **GNS** needs platform v. 2.0.

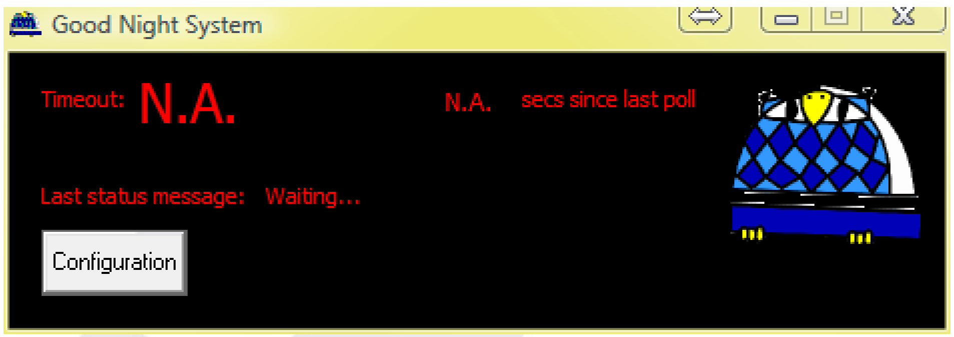
Apart from the main software itself, in the installation folder (c:\program files\GNS, or c:\program files (x86)\GNS, the latter if yours is a 64 bit system), you'll find a few scripts (small programs):

- update.vbs
- alarm.vbs
- switchoff.vbs
- idle.vbs
- message.vbs

These scripts will be used to integrate the **GNS** with some automation systems in the market – other systems have or may have native support.

- Now run the software (there's a shortcut from your start menu), and you'll see the main window:

[1]If you plan to use the system with Maxpilote, you'll have to select the Maxpilote folder for installation. If you plan to use SGPro, you should uninstall the GNS after the tests, as SGPro includes it's own version.

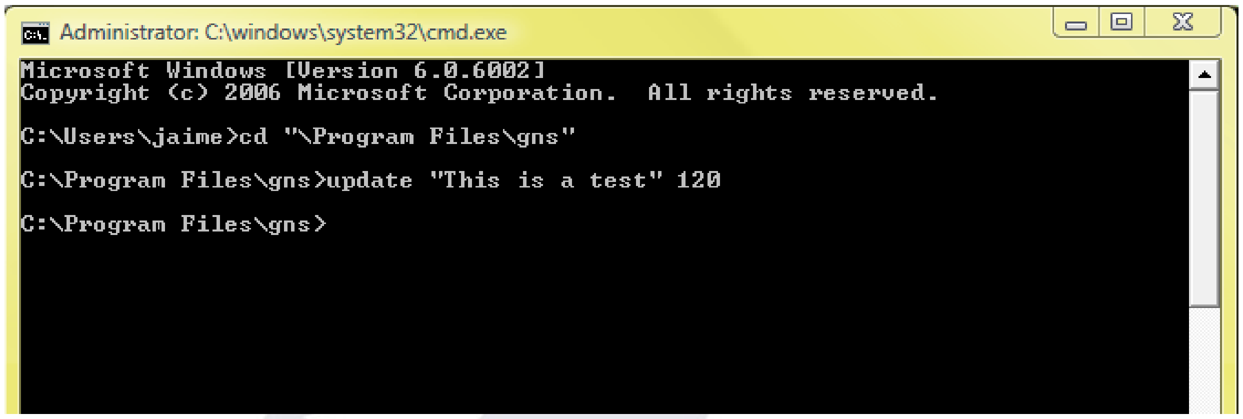


FIREWALL

VERY IMPORTANT AT THIS POINT: if you have any kind of firewall installed, it will warn you of the "Good Night System" program (GNS.exe) trying to communicate with the outer world. This is normal, since the GNS must start listening for incoming requests, and you should enable (unblock) it in order for the system to work.

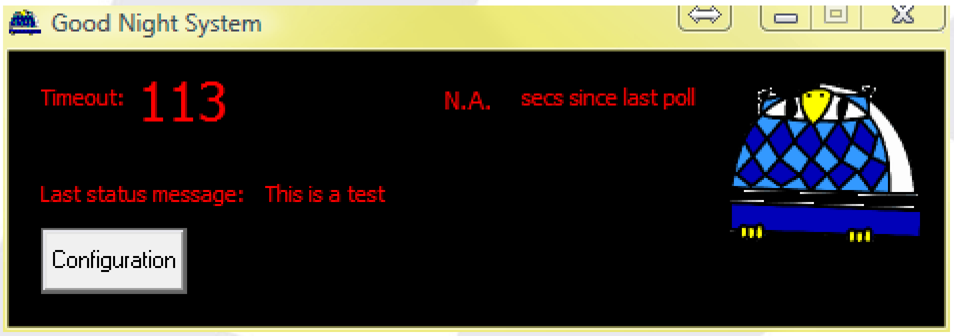
- Let's confirm the installation went well:
  - Run a command prompt:
    - Start menu → All programs → Accessories → Command prompt
  - Go to the installation folder:
    - *For 32 bits systems,* type: **cd "\program files\gns"** including the quotes and followed by the return key
    - *For 64 bits systems*, type: **cd "\program files (x86)\gns[2]"** including the quotes and followed by the return key
  - Now type: **update "This is a test" 120** followed by return
  - Your command window should look like this:

[2]Even if the GNS is an "Any CPU" application, it will install in the 32 bit folder for 64 bits system.



```
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\jaime>cd "\Program Files\gns"

C:\Program Files\gns>update "This is a test" 120

C:\Program Files\gns>
```

– And check the **GNS** window did actually update, to show (7 seconds later...):



```
Timeout: 113                    N.A.   secs since last poll

Last status message:   This is a test
```

Note the status message has been updated, … and verify it will count down to 0.

"secs since last poll" remains at **N.A**. as the program has never been contacted by a smartphone; this field will let us know how many seconds have elapsed since the smartphone last requested an update.

Do not close the "cmd" window as we'll use it later.

Please forget the configuration button for the moment.

3. Smartphone installation

Let's go now to the smartphone:

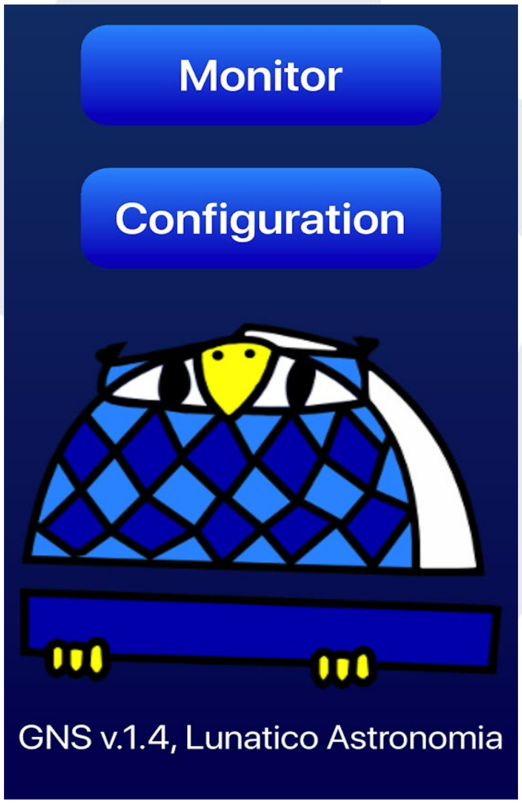
Enable your smartphone network connection (be it wifi or mobile data/3G/4G/35G, whatever).

Download our Free app (from *Google play* for **Android phones**, or from the *AppStore*, in case of **iPhone**).

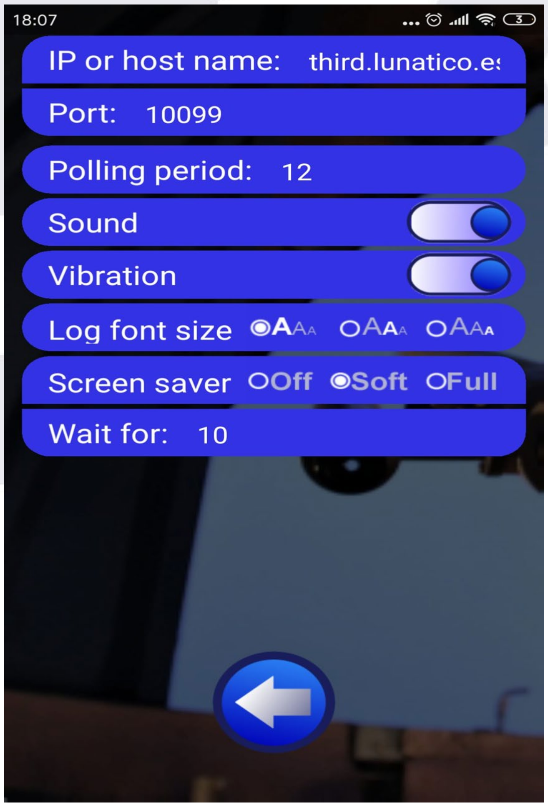
Is is highly recommended to try with the <u>free version</u> first; it is almost identical to the full version, with only a couple of exceptions:
– the watchdog will be disabled and will act as a countdown timer, limiting the monitored session at 30 minutes.
– the length of the messages displayed is shorter

This way you'll be sure everything will run smooth before spending money in the paid version.

Run the app, and you will get to the initial screen:



Press "Configuration" to open the parameters screen:



18:07

IP or host name: third.lunatico.es

Port: 10099

Polling period: 12

Sound [✓ On]

Vibration [✓ On]

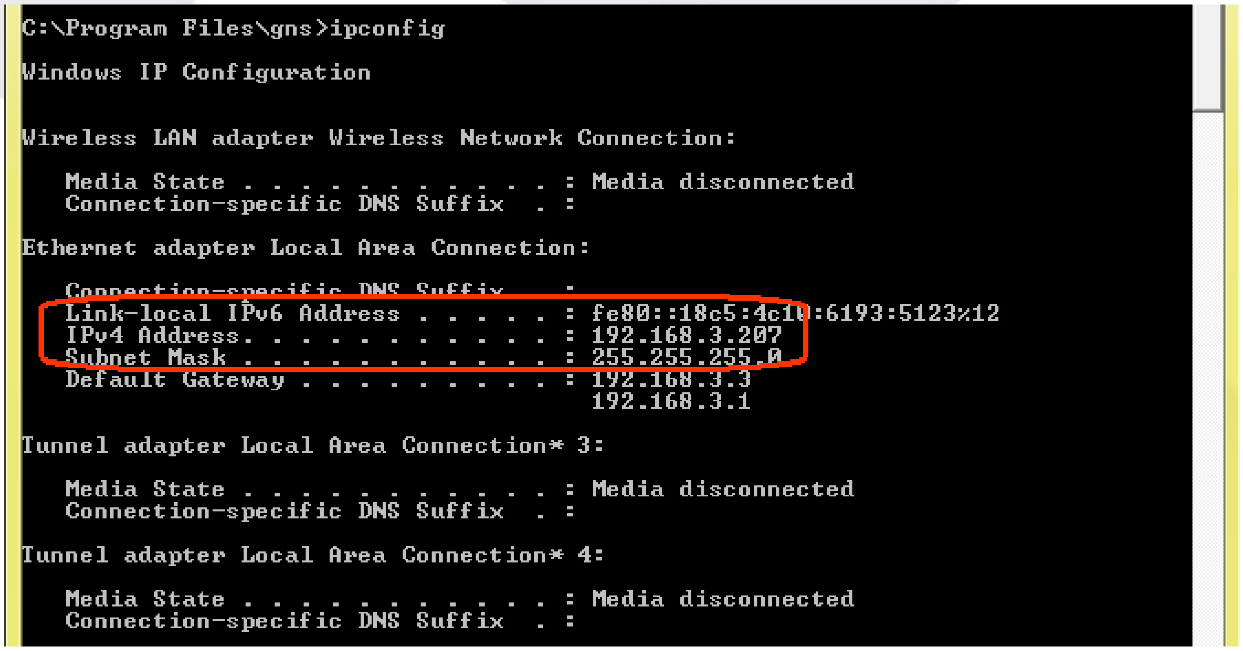
Log font size [✓ AAA] AAA AAA

Screen saver Off [✓ Soft] Full

Wait for: 10

At this point, we have to get this app communicating with our PC. Please type the host name (or just the IP address if you are in the same network) of your observatory PC. This can be a bit difficult if you know nothing about networks, but having a remote or automated observatory you most likely know enough.

In case of doubt, go back to your PC, cmd window, and type **ipconfig** followed by return, and look for a meaningful (!) set of 4 dot separated numbers, in a line that will read something like “IPv4 address” (see below):



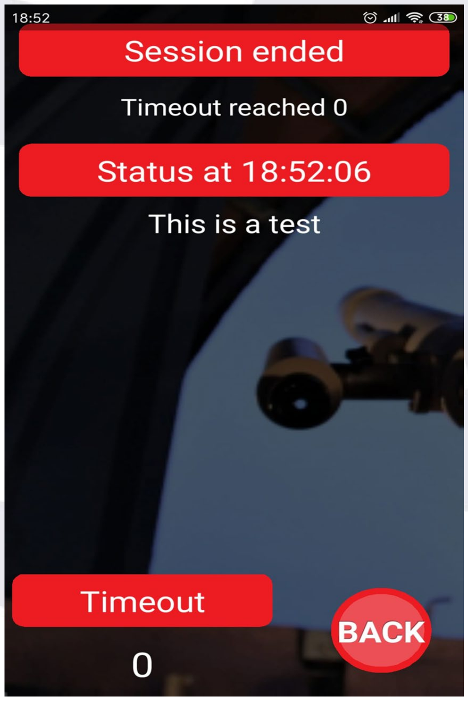
```
C:\Program Files\gns>ipconfig

Windows IP Configuration


Wireless LAN adapter Wireless Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::18c5:4c10:6193:5123%12
   IPv4 Address. . . . . . . . . . . : 192.168.3.207
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.3.3
                                       192.168.3.1

Tunnel adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 4:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

For the time being, leave the rest of fields as they are:

- **Port**: is the tcp/ip "slot" where the requests will be sent (we can change this in the windows application, configuration, in the event the default port is already in use).
- **Polling period**: is the number of seconds the smartphone app will wait after an update to check again.
- **Sound** and **Vibration**, just refer to the means of warning you. Leave at least one active!
- **Log font size**, allows you to adjust the size of the messages displayed in the monitor window.
- **Screen saver**, especially for OLED displays, as they get damaged if the image is static for a long time. Off, Soft (will enable the display after a message is received), or Full – will only become active when the screen is touched.
- **Wait for**, the seconds the screen saver will wait to become active.

Just touch the back button, and from the main screen now press "**Monitor**".

If the network is correctly setup, firewall included, we'll go to the main monitor screen, and in a few seconds the communications will go "UP", and a forced alarm will fire. It is forced because, unless you performed all these steps in less than 2 minutes (we setup a 120 sec time earlier, testing the PC application), the timeout reached 0 long ago.



Timeout reached 0

This is a test

What if the communications don't go UP and you get other kind of error (watchdog, or "too many conn tries")?

Well, that means the smartphone is not able to talk to the PC. There are a number of possible causes:

– some firewall program is preventing the PC to accept incoming communications; completely disable the firewall and try again. If this works, you should enable the firewall later, but adding a rule, or exception, for the GNS.exe program.

– The network settings are not correct. This is a more complex matter. I see three different situations:

a) your have a backyard observatory (or otherwise at home) that shares the same network that the smartphone. You usually control your observatory PC from your living room, using Windows remote desktop or similar software.

This is the most difficult case, as your PC is probably getting it's IP address using DHCP (a network protocol that assigns a different address every time you power on); that is, in your Windows network settings you've leave the "IP v4 address" setting at the default "automatic".

Using the "ipconfig" program you can find out the current IP address (the 4 number set), but it can change every session, maybe even during an imaging sessions (the address has an expiry time), and this can ruin our purpose. Please do assign your PC a fixed IP address. I'll try to explain how to do this in future updates to this manual, but for the moment please seek assistance if you don't know how to do it.

If this is your case, set the "Host" field of the smartphone configuration to your 4 digit IP address.

b) you have a remote observatory, connecting to it via Internet and using a program such as teamviewer or similar.

In this case the settings are a bit more complex, but you already know or have someone who knows. In a nutshell, you'll need:

– a fixed internal network IP address in your observatory PC (needed for the second step).
– a port "opened" in your observatory router (as to route incoming 10099 TCP port messages to the 10099 port of your PC)
– if your router has a dynamic IP (that is, the public IP address changes from time to time, and every time you power on the router), then you need a dynamic host name, a service provided for free by several network services (such as dyndns.org).

c) your observatory is at home, but in a different network than your mobile.

This is the same, from our point of view, as case (b).

The point of all this is being able to configure your smartphone so its messages will reach the destination PC.

> ***If b or c is your case, please type your complete host name (similar to "myobservatory.dyndns.org") in the "Host" filed of the smartphone configuration.***

4. Real life integration

You have surely understood the way it works now; we are going to update the system with every action (or groups of actions, more likely) the scope is going to do, allotting time for them. If the time expires, then an alarm will be issued.

We can also issue an immediate alarm, and of course notify the system of a successful end of the session.

In the simplistic example of the web site – to be honest reflects the approach I'm actually using, and works great – I just update the **GNS**:

- once at the start of the session, with a 10 minute timeout. In this time main imaging should have been reached.
- Once for every imaging group (including telescope slew, plate solve, focus, and actual imaging). The duration of course varies, up to a few hours.
- Last for the end of the session: ccd warm up, telescope park, roof close.

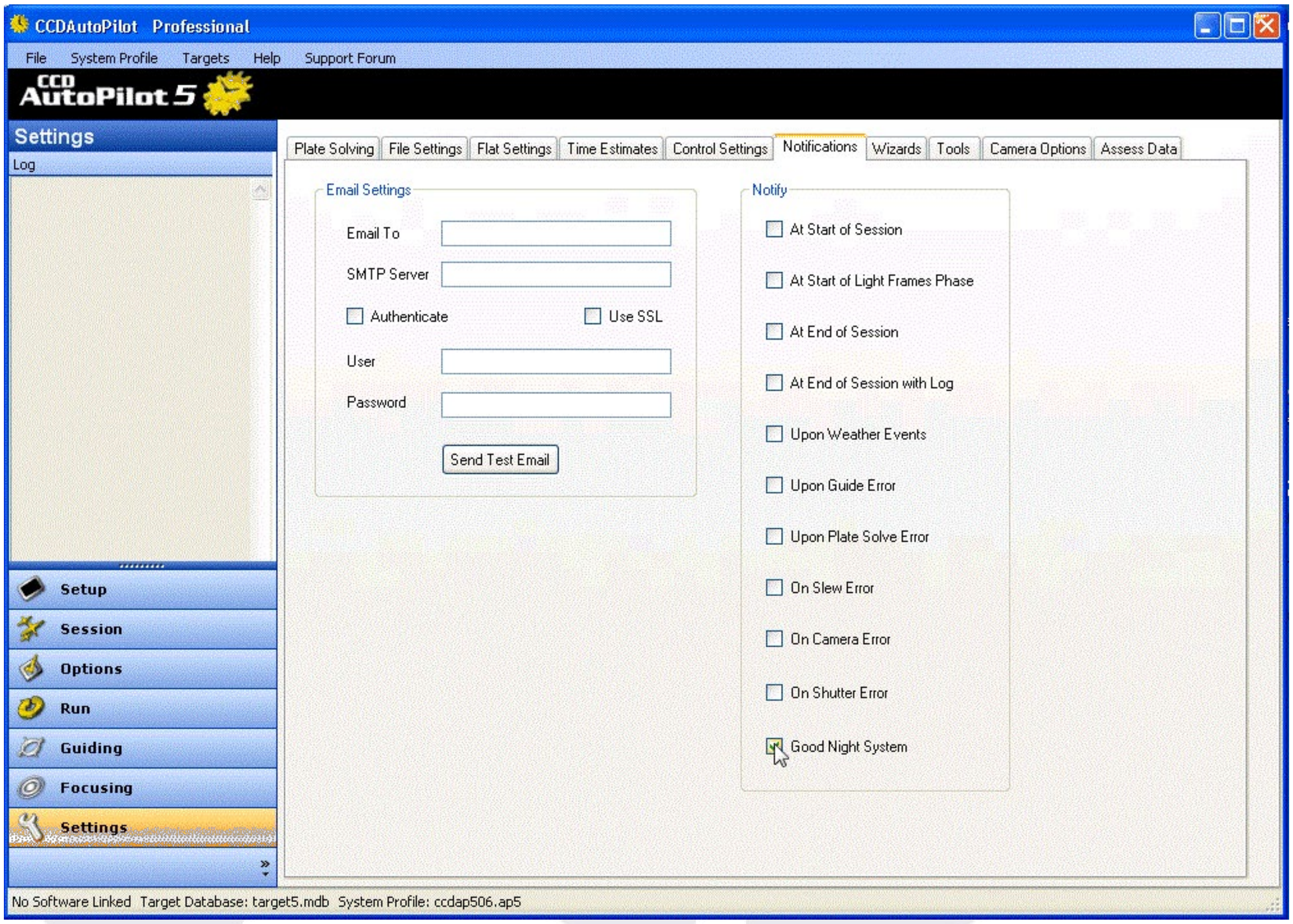
As my observatory is at home, I also launch an immediate alarm in case of unsafe weather detected by my CloudWatcher, but this is a bit of a lazy approach.

At the moment of this writing, the **GNS** can be used with 6 popular session control systems: *CCDAutoPilot, Astro Photography Tool, Sequence Generator Pro* and *Maxpilote* support the system directly, so no work is needed from your part. Just take a look at next section to see any installation / configuration notes.

CCDCommander and *ACP* can be used quite easily too, but require you to use their scripting features; please see instructions below.

4.1. CCDAutoPilot 5

As already mentioned, CCDAP v5 includes native support for the GNS, so you'll just have to activate it from the Settings window, Notifications tab:



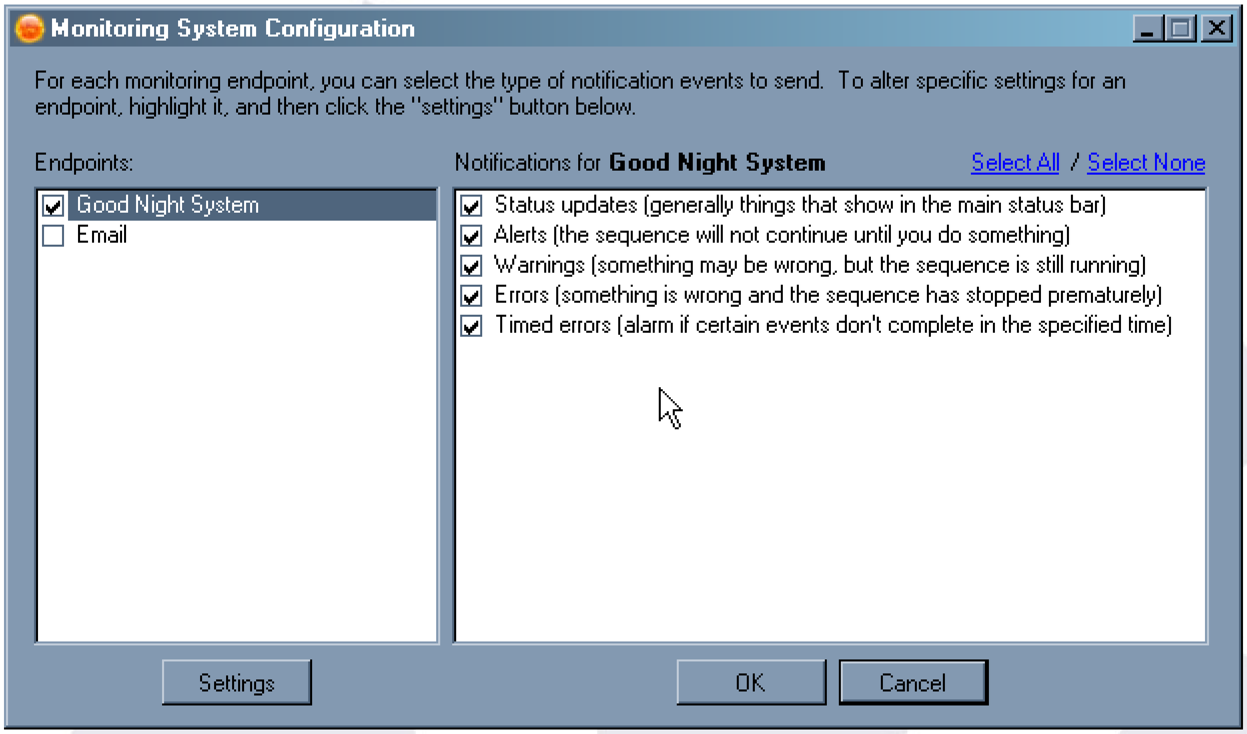
- [ ] At Start of Session
- [ ] At Start of Light Frames Phase
- [ ] At End of Session
- [ ] At End of Session with Log
- [ ] Upon Weather Events
- [ ] Upon Guide Error
- [ ] Upon Plate Solve Error
- [ ] On Slew Error
- [ ] On Camera Error
- [ ] On Shutter Error
- [✓] Good Night System

No Software Linked Target Database: target5.mdb System Profile: ccdap506.ap5

And that's all, the GNS will be launched by CCDAP and your smartphone will track your session!

4.2. Sequence Generator Pro

Main sequence's software SGPro has an add-on available to support notifications. To enable GNS, you just have to enable it (from the **Tools** menu, **Configure notifications** option):



For each monitoring endpoint, you can select the type of notification events to send. To alter specific settings for an endpoint, highlight it, and then click the "settings" button below.

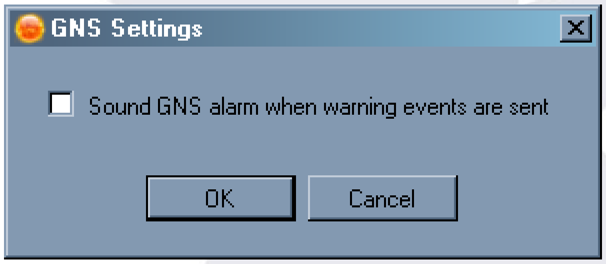
Endpoints:

- [✓] Good Night System
- [ ] Email

Notifications for **Good Night System** Select All / Select None

- [✓] Status updates (generally things that show in the main status bar)
- [✓] Alerts (the sequence will not continue until you do something)
- [✓] Warnings (something may be wrong, but the sequence is still running)
- [✓] Errors (something is wrong and the sequence has stopped prematurely)
- [✓] Timed errors (alarm if certain events don't complete in the specified time)

There's even an additional setting to locally play a sound if a warning is sent to the GNS:

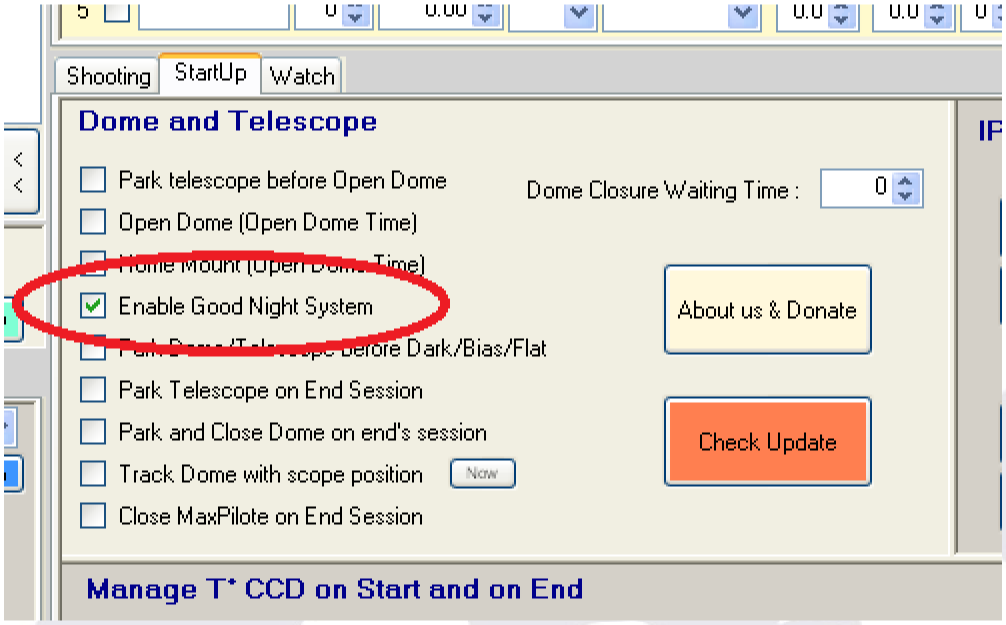


- [ ] Sound GNS alarm when warning events are sent

Important: SGPro already installs its own copy of the GNS software. Do not install it independently, if you already did, just uninstall the GNS from the PC before using SGPro's version.

4.3. Maxpilote

Maxpilote's author has already added native support for the Good Night System:



Dome and Telescope

- [ ] Park telescope before Open Dome
- [ ] Open Dome (Open Dome Time)
- [ ] Home Mount (Open Dome Time)
- [✓] Enable Good Night System
- [ ] Park Dome/Telescope before Dark/Bias/Flat
- [ ] Park Telescope on End Session
- [ ] Park and Close Dome on end's session
- [ ] Track Dome with scope position
- [ ] Close MaxPilote on End Session

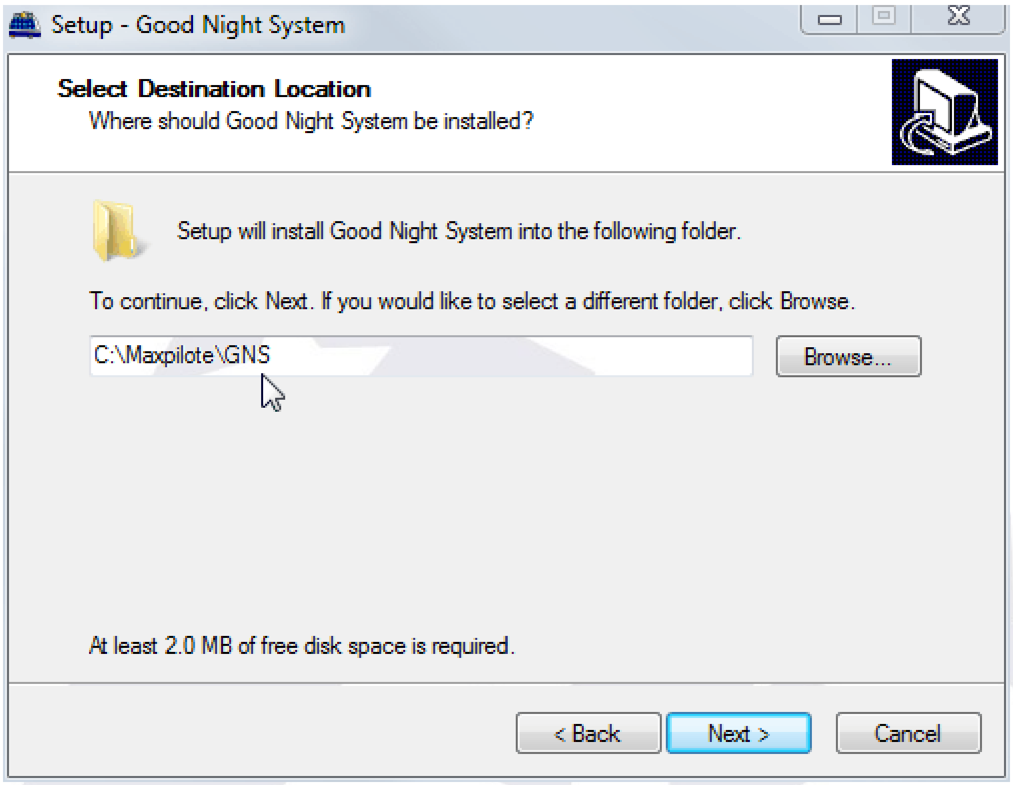
Dome Closure Waiting Time : 0

Manage T° CCD on Start and on End

... while I've been unable to test it fully, I'm sure any possible issues will be addressed by him the great way Maxpilote users are used to.

Very important: for the GNS to work with Maxpilote, the software must be installed in the same folder as Maxpilote.

Please check as the installation program will try to install it under "Maxpilote\GNS" instead.



Setup - Good Night System

Select Destination Location

Where should Good Night System be installed?

Setup will install Good Night System into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

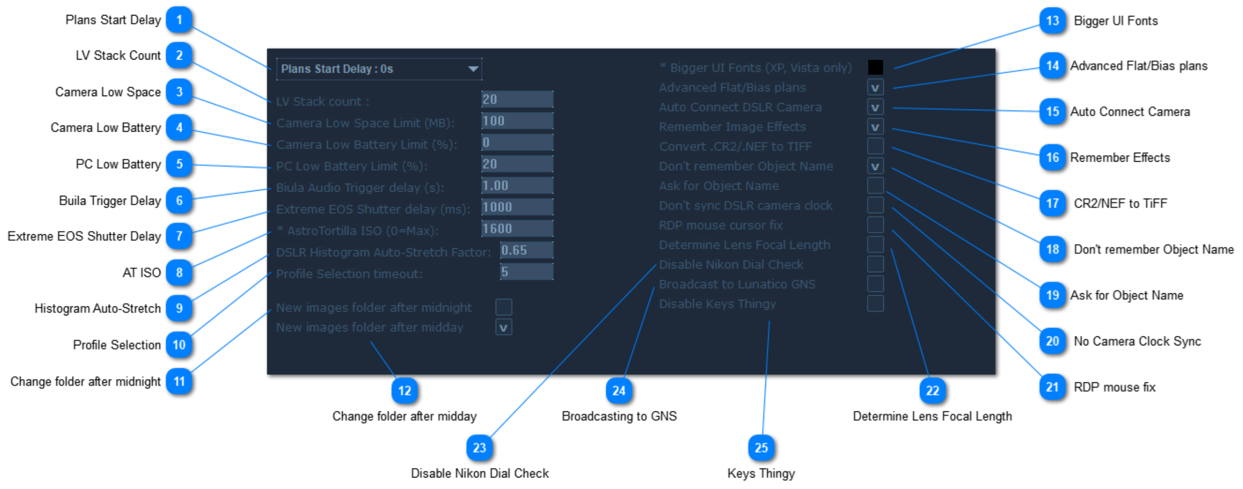
C:\Maxpilote\GNS

At least 2.0 MB of free disk space is required.

... you'll have to manually remove the trailing "\GNS" from the installation folder.

4.4. APT (astro photography tool)

APT also offers GNS integration natively:



Plans Start Delay : 0s

LV Stack count :	20
Camera Low Space Limit (MB):	100
Camera Low Battery Limit (%):	0
PC Low Battery Limit (%):	20
Biula Audio Trigger delay (s):	1.00
Extreme EOS Shutter delay (ms):	1000
* AstroTortilla ISO (0=Max):	1600
DSLR Histogram Auto-Stretch Factor:	0.65
Profile Selection timeout:	5
New images folder after midnight	☐
New images folder after midday	v

* Bigger UI Fonts (XP, Vista only)	■
Advanced Flat/Bias plans	v
Auto Connect DSLR Camera	v
Remember Image Effects	v
Convert .CR2/.NEF to TIFF	☐
Don't remember Object Name	v
Ask for Object Name	☐
Don't sync DSLR camera clock	☐
RDP mouse cursor fix	☐
Determine Lens Focal Length	☐
Disable Nikon Dial Check	☐
Broadcast to Lunatico GNS	☐
Disable Keys Thingy	☐

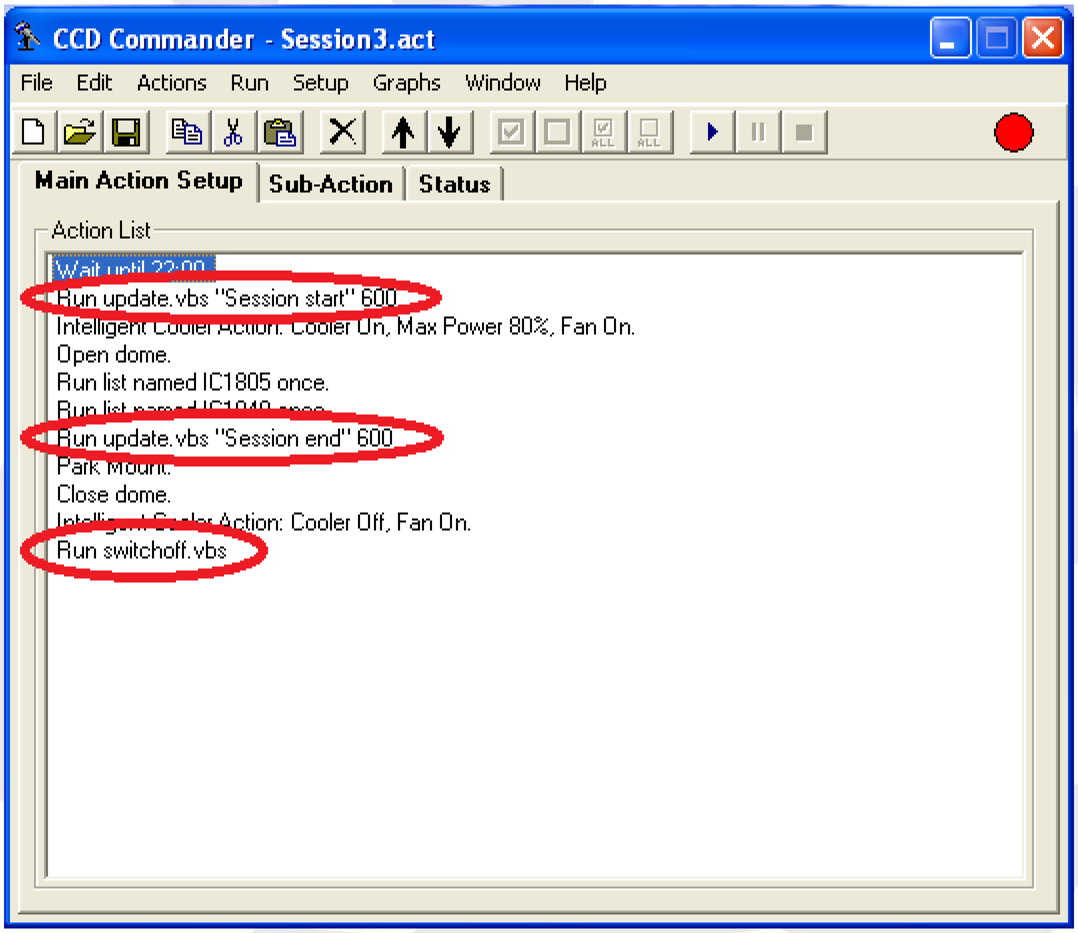
Setting number 24 allows you to activate GNS notifications.

Couldn't be easier.

4.5. CCDCommander

Integrating this system with CCDCommander is really straighforward, as CCDC allows the user to call scripts in many cases and at any point of the action list.

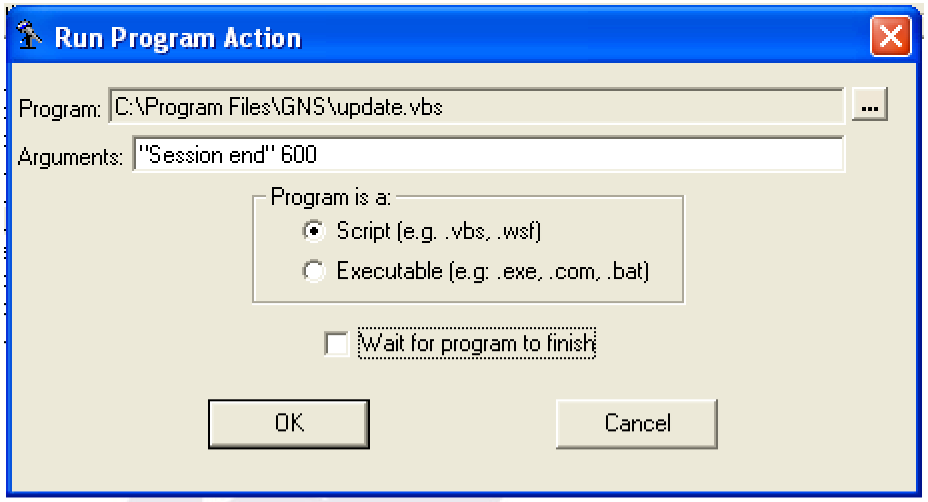
At selected points in your action list, insert a "Call external subprogram" action, to execute the script "update.vbs" included with your **GNS** distribution. You'll have to call it with two parameters, the first one being an informative message, the second one the timeout in seconds. A simple example:


CCD Commander - Session3.act
File Edit Actions Run Setup Graphs Window Help
Main Action Setup
Sub-Action
Status
Action List

```
Wait until 22:00
Run update.vbs "Session start" 600
Intelligent Cooler Action: Cooler On, Max Power 80%, Fan On.
Open dome.
Run list named IC1805 once.
Run list named IC1848 once.
Run update.vbs "Session end" 600
Park Mount.
Close dome.
Intelligent Cooler Action: Cooler Off, Fan On.
Run switchoff.vbs
```

As can be seen, in the "Main Action Setup" I just added two calls to "Update", one at the beginning of the session, another when we start winding up, and a final one to disable the system ("Switchoff").

For each call you'll use the "Run external program" option from the "Actions" menu:


Run Program Action

Program: C:\Program Files\GNS\update.vbs

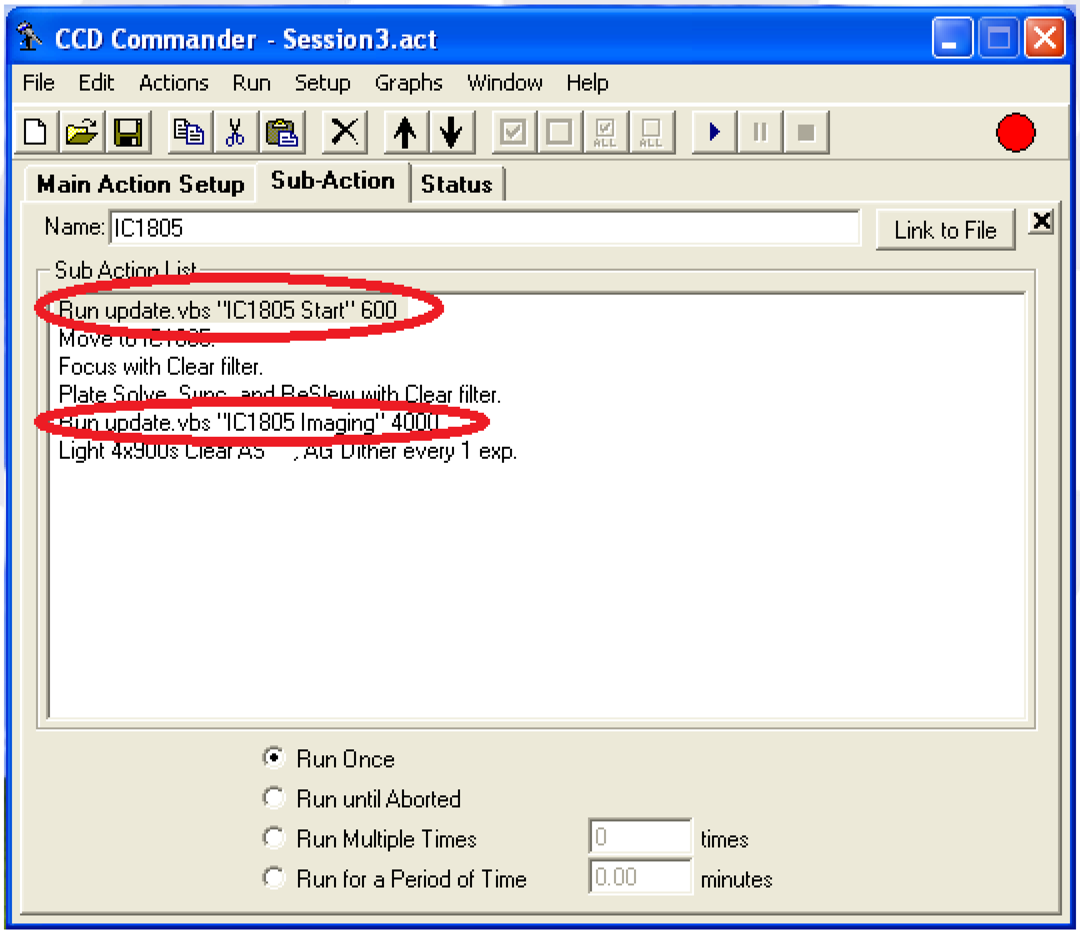
Arguments: "Session end" 600

Program is a:
[✓ Script (e.g. .vbs, .wsf)]
Executable (e.g: .exe, .com, .bat)

Wait for program to finish

If the message includes more than one word, please enclose it in quotes as in "Session end". Also please note we've selected "Program is a... script", and unchecked (not really important) the optional "Wait for program to finish".

Next, in each Sub-action, you can add as many updates as you wish:


CCD Commander - Session3.act

Name: IC1805

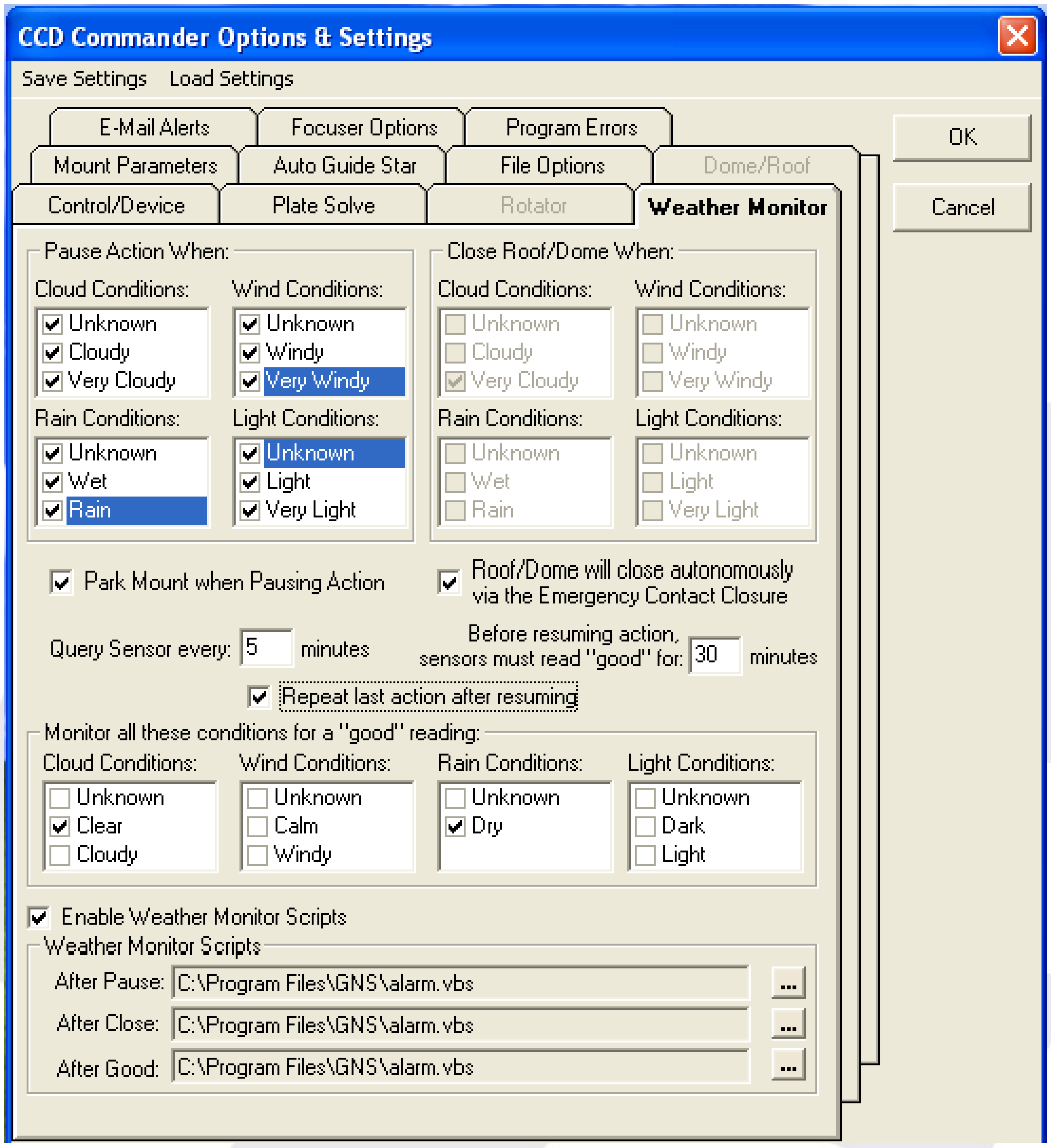
Sub Action List

```
Run update.vbs "IC1805 Start" 600
Move to IC1805.
Focus with Clear filter.
Plate Solve, Sync, and ReSlew with Clear filter.
Run update.vbs "IC1805 Imaging" 4000
Light 4x900s Clear AS     , AG Dither every 1 exp.
```

[✓ Run Once]
Run until Aborted
Run Multiple Times 0 times
Run for a Period of Time 0.00 minutes

Don't forget to end the main action list calling the script "switchoff.vbs"; this one will inform your smartphone the session ended successfully, and it will stop the communication.

You may also want to call “alarm.vbs” in case of errors, or maybe in case of unsafe weather detected (note: alarm is just an update with a timeout of 0 seconds).



Pause Action When:

Cloud Conditions: [✓ Unknown] [✓ Cloudy] [✓ Very Cloudy]
Wind Conditions: [✓ Unknown] [✓ Windy] [✓ Very Windy]
Rain Conditions: [✓ Unknown] [✓ Wet] [✓ Rain]
Light Conditions: [✓ Unknown] [✓ Light] [✓ Very Light]

Close Roof/Dome When:

Cloud Conditions: Unknown, Cloudy, [✓ Very Cloudy]
Wind Conditions: Unknown, Windy, Very Windy
Rain Conditions: Unknown, Wet, Rain
Light Conditions: Unknown, Light, Very Light

[✓ Park Mount when Pausing Action]
[✓ Roof/Dome will close autonomously via the Emergency Contact Closure]

Query Sensor every: 5 minutes
Before resuming action, sensors must read "good" for: 30 minutes
[✓ Repeat last action after resuming]

Monitor all these conditions for a "good" reading:

Cloud Conditions: Unknown, [✓ Clear], Cloudy
Wind Conditions: Unknown, Calm, Windy
Rain Conditions: Unknown, [✓ Dry]
Light Conditions: Unknown, Dark, Light

[✓ Enable Weather Monitor Scripts]

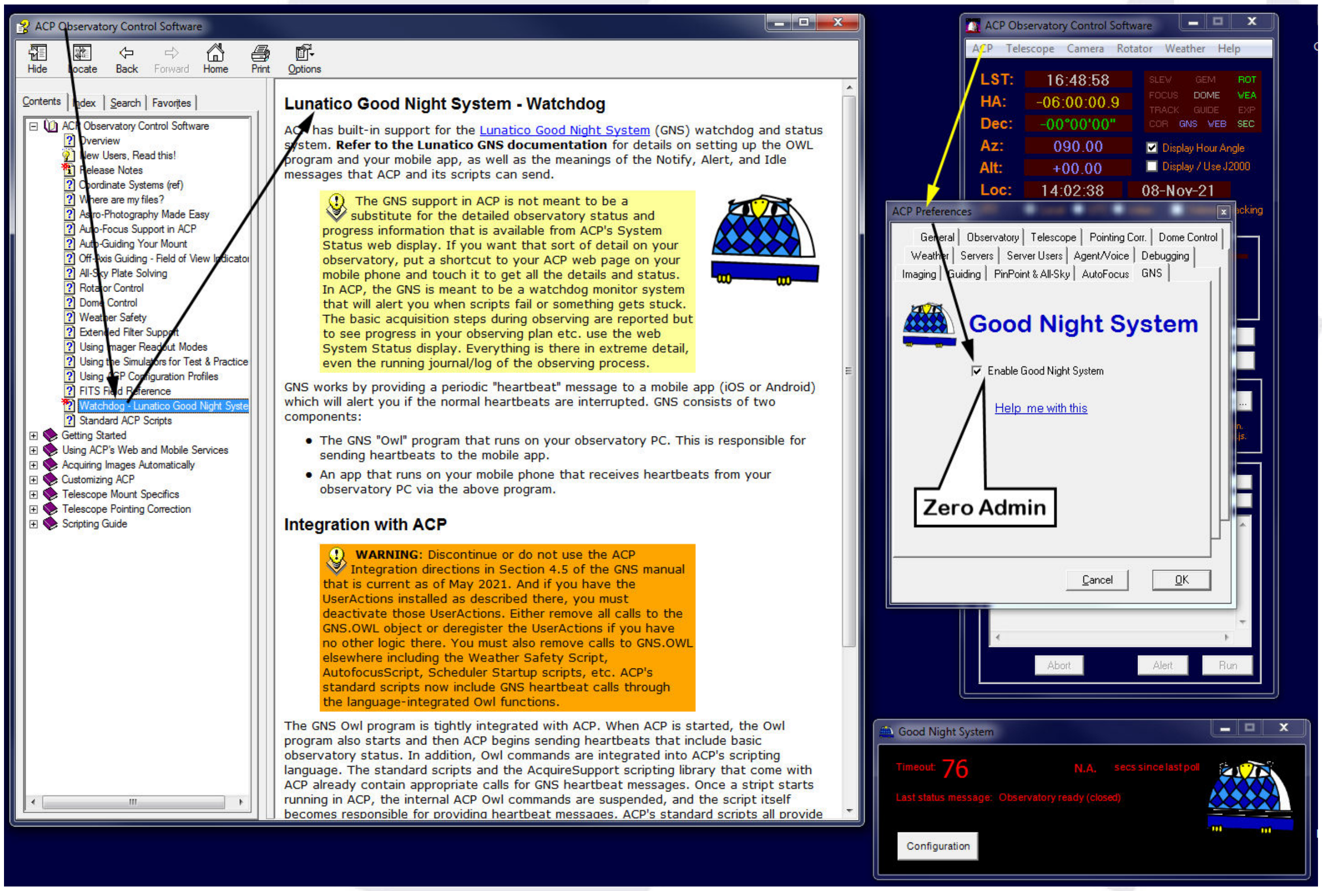
Weather Monitor Scripts

After Pause: C:\Program Files\GNS\alarm.vbs
After Close: C:\Program Files\GNS\alarm.vbs
After Good: C:\Program Files\GNS\alarm.vbs

4.6. ACP Observatory Control Software

Please note: From ACP Expert version 9, the integration of GNS has been simplified to a single click on the GNS Preference Tab. GNS support is now built into ACP's own scripts and the vast majority of users should have no need to write the UserActions previously detailed in this manual.

To avoid confusion, this section of the manual has been removed. Should you require a copy of the old manual with the detailed script descriptions, please get in touch with us at support@lunaticoastro.com.



Lunatico Good Night System - Watchdog

ACP has built-in support for the Lunatico Good Night System (GNS) watchdog and status system. **Refer to the Lunatico GNS documentation** for details on setting up the OWL program and your mobile app, as well as the meanings of the Notify, Alert, and Idle messages that ACP and its scripts can send.

The GNS support in ACP is not meant to be a substitute for the detailed observatory status and progress information that is available from ACP's System Status web display. If you want that sort of detail on your observatory, put a shortcut to your ACP web page on your mobile phone and touch it to get all the details and status. In ACP, the GNS is meant to be a watchdog monitor system that will alert you when scripts fail or something gets stuck. The basic acquisition steps during observing are reported but to see progress in your observing plan etc. use the web System Status display. Everything is there in extreme detail, even the running journal/log of the observing process.

GNS works by providing a periodic "heartbeat" message to a mobile app (iOS or Android) which will alert you if the normal heartbeats are interrupted. GNS consists of two components:

- The GNS "Owl" program that runs on your observatory PC. This is responsible for sending heartbeats to the mobile app.
- An app that runs on your mobile phone that receives heartbeats from your observatory PC via the above program.

Integration with ACP

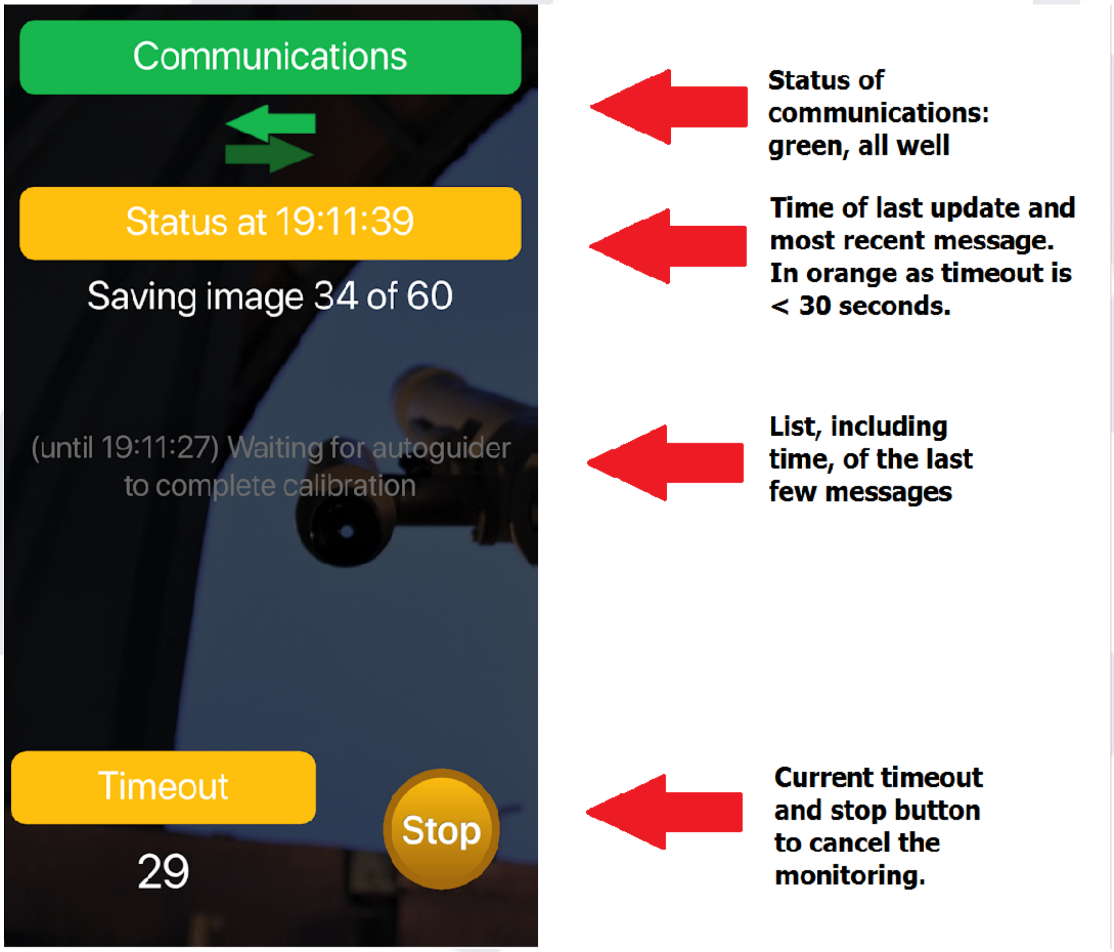
WARNING: Discontinue or do not use the ACP Integration directions in Section 4.5 of the GNS manual that is current as of May 2021. And if you have the UserActions installed as described there, you must deactivate those UserActions. Either remove all calls to the GNS.OWL object or deregister the UserActions if you have no other logic there. You must also remove calls to GNS.OWL elsewhere including the Weather Safety Script, AutofocusScript, Scheduler Startup scripts, etc. ACP's standard scripts now include GNS heartbeat calls through the language-integrated Owl functions.

The GNS Owl program is tightly integrated with ACP. When ACP is started, the Owl program also starts and then ACP begins sending heartbeats that include basic observatory status. In addition, Owl commands are integrated into ACP's scripting language. The standard scripts and the AcquireSupport scripting library that come with ACP already contain appropriate calls for GNS heartbeat messages. Once a stript starts running in ACP, the internal ACP Owl commands are suspended, and the script itself becomes responsible for providing heartbeat messages. ACP's standard scripts all provide

5. Important smartphone usage notes

– leave your smartphone fully charged or plugged to a suitable power supply – it won't warn you if the battery runs out!
– dim the screen to save battery
– check your sound settings if you are using the audible alarm
– **the app won't run in the background!** Communications are not reliable at all while in the background, so it is mandatory to leave the app running, the monitor screen visible. In fact, for iOS devices, if the app is "sent to the background", it will stop.

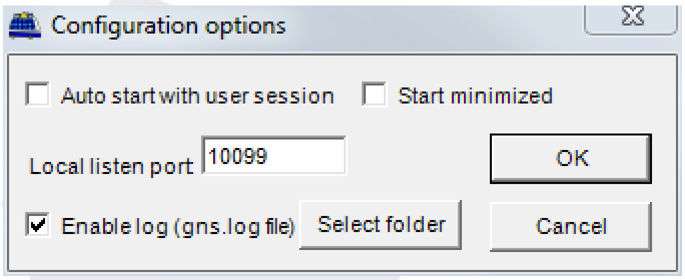
Just in case, here's an explanation of everything in the monitor screen:


Communications
Status at 19:11:39
Saving image 34 of 60
(until 19:11:27) Waiting for autoguider to complete calibration
Timeout
29
Stop
Status of communications: green, all well
Time of last update and most recent message. In orange as timeout is < 30 seconds.
List, including time, of the last few messages
Current timeout and stop button to cancel the monitoring.

The timeout shown in the smartphone will typically be a few seconds delayed compared with the one in the PC; this is normal.

6. Windows application options

Before going on, let's take a look at the few available options in the windows program:



Auto start with user session: [ ]
Start minimized: [ ]
Local listen port: 10099
Enable log (gns.log file): [✓]

Auto start with user session: if checked, the GNS program will be launched and will start accepting connections as soon as you log in windows. May be useful if you prefer to have your smartphone connected even before the real session has started.

Start minimized: if checked, the application will be "hidden" in the system tray, to avoid cluttering your desktop, every time it is launched. Of course you can make it visible again just clicking its icon in the tray.

Local listen port: this is the "slot" the program will be awaiting connections from. It must match the one selected in the smartphone app. We chose 10099 as ports above 10000 are freely available to programs; no problem in changing it if you have that port already in use, but please check the available ports in your smartphone app.

If you change the port, you'll have to restart the application for the change to take effect.

Enable log file: will create (or append to if already exists) a text file describing the session events, from the point of view of the windows software. Should be useful in finding out problems. Of course the "**Select folder**" button will do just as its name implies.

Don't forget to remove the file from time to time, as it can grow big.

Last, you are not limited to the supplied scripts (even if everything can be done with just the "update.vbs" one); should you find it useful, just edit any of them with a suitable editor (the standard "notepad" will do nicely) to issue personalized messages easily.

7. Tight integration

If you are a developer, either of your own, or a commercial or open source or whatever CCD session automation program, here's a short guide on how to integrate GNS into your system.

Basically you (the imaging/session controller) send messages to the system in certain cases:

- to announce a new task, with a descriptive name and a timeout (you can change name or timeout at any moment)
- to raise an alarm
- to suspend the system temporarily
- to end the session

Ideally, the system, from the start, will be getting updates of tasks before the timeout is reached. Ideally, again, for perfection, the system is never paused during the session - just disabled at the end.

The user will get an alarm if:

- the communication PC / Smartphone fails (comms problems, or windows freezing...)
- any task times out
- your program issues such an alarm

This way he is fairly sure everything's running smoothly, apart from getting some information on the status.

The pc program and the app are in contact via a TCP socket - so it is real time communication.

You can update the pc program either via COM (there's an object, GNS.OWL, with a few methods), or executing a vbs script that in turn calls the COM object.

When you install the software, there are a few such scripts in the install folder.

If (as I expect) you'd rather use the COM interface, the methods are just 3:

NewMsg: takes a string argument, and just updates the displayed Status

NewTimeout: takes a long integer argument, and updates the timeout:

– if *timeout > 0*, the behavior is normal
– if *timeout = 0*, then an alarm is issued as soon as it is received by the

smartphone
– if *timeout = -1*, then the count down is stopped, and the system is “waiting” (but the watchdog is still active!)
– if *timeout = -2*, then a switch off is performed (it is understood the end of the session has been reached).

CurrentTimeout: no arguments, returns a long integer with the current timeout according the the GNS.

Once you start the communication with the GNS, its status window will appear but can go directly hidden to the notification area if the user so chooses.

There is a Visual Studio Vbasic .NET very simple example project here: https://lunaticoastro.com/GNS/GNSConn.zip

So, a minimalist session would go (from the point of view of the automation program):

- session starts → send to GNS "Starting session" 300
- cooling down CCD -> send to GNS "Cooling down" 120
- acquiring first target -> send to GNS "Slewing to IC434" 60
- take image, filter R -> send to GNS "R filter, 600 sec expo, IC434" 800
repeat...
- parking mount -> send to GNS...
- closing roof ->...
- end of sesion -> GNS close session

The key points are:

– the timeouts can be generous, no need to risk a false alarm. We just want to detect if something happens to your software, windows (more likely), whatever, that prevents the next update. But, in general, this is not urgent, a few minutes will make no difference

– the GNS should be active during the whole session - no pauses. While paused, its protection is effectively disabled. Would be great to avoid this.

– the status message can be informative, but if too long won't be so easily read. I'd avoid for example file names in the messages. In general 60 to 80 char long messages are displayed nicely in every phone.

I'm hope this explanation is detailed enough, but of course I'll be glad to help.

8. Pending features and wishlist

As of today, the “wish list” has been reduced to very short one:

Smartphone:

– allow the user to set the volume in the configuration screen

System / General:

– add another status message, so an additional program can check, for instance, the weather, roof status, mount parked status and report it to the smartphone.

9. Last notes

I'm confident you understand there's no warranty, whatsoever. Use the software at your own risk. If your expensive equipment gets damaged for any reason, do not blame us.

Developing the system has taken much longer than expected, for a variety of reasons, but it's been worth the effort.

Acknowledgments

Dave Randall, who basically redesigned the monitor window in a clear, nice, concise way.

Yvo (Yvalo Stoynov), for his great integration into **APT.**

Laurent Bourgon, for taking the time to add support for the **GNS** in his Maxpilote software.

Ken and **Jared**, authors of Sequence Generator Pro, also for supporting this system in their software.

John Smith, author of CCDAutoPilot, for his nice integration of the **GNS** in his software.

Matt Thomas, author of CCDCommander, for allowing me to post the messages about the system in his support list.

… and of course everybody in the beta test team for their contributions and patience.

Revision history:

- 1.0 Original manual
- 1.0.0.1 More clear 32/64 bits explanations, more detailed Maxpilote installation, updated everything related to CCDAP v5. New "message.vbs" script.
- 1.0.0.2 Bigger space for messages, updated for SGPro's support, local log file creation.
- 1.0.1.0 Much neglected update. Developer section.
- 1.0.1.1 Changes to ACP implementation.