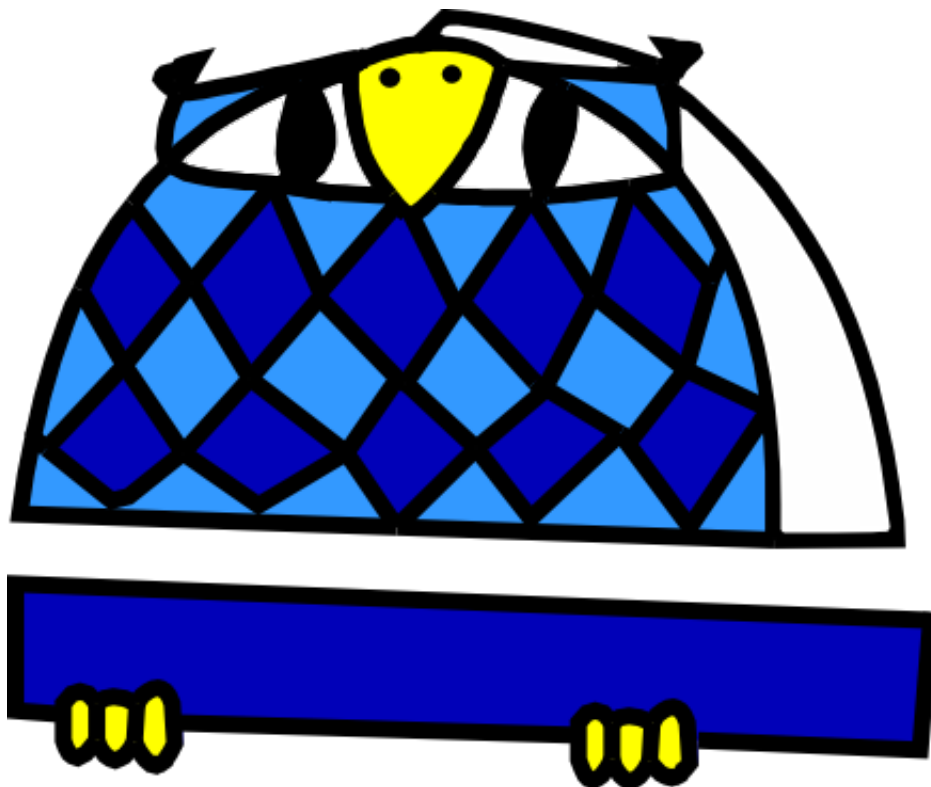


GNS

Good Night System

User's manual



By Lunático Astronomía

INDEX

Preface.....	3
Installing and checking the PC program.....	4
Installing the software.....	4
Smartphone installation.....	6
If you got an error.....	9
Real life integration.....	11
CCDAutoPilot 5.....	12
Sequence Generator Pro.....	12
Maxpilote.....	13
APT (astro photography tool).....	15
ACP Observatory Control Software.....	15
N.I.N.A.....	16
CCDCommander.....	17
Important smartphone usage notes.....	19
Windows application options.....	20
Tight integration.....	21
Pending features and wish-list.....	23
Smartphone.....	23
System / General.....	23
Last notes.....	24
Warranty (or lack thereof!).....	24
Acknowledgments.....	24
Revision history.....	24

Preface

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

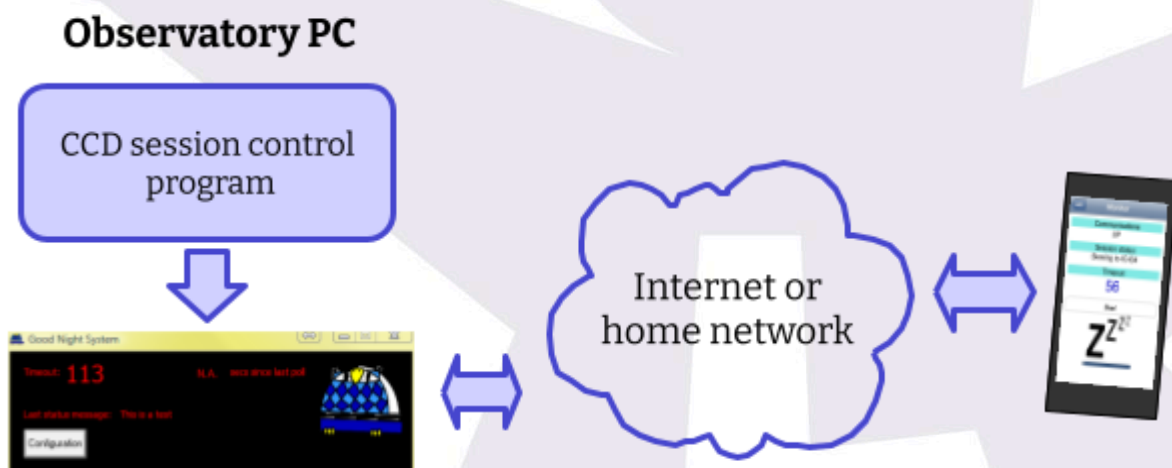
These are some features that set apart the Good Night System:

- it **does not rely on SMSs or emails**; to the contrary, the smartphone constantly plays an active role.
- safety comes 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 small windows program
- a smartphone app

It will work as shown in the following drawing:



So on one side, we'll have our PC GNS program being updated by our ccd/session control software of choice. On 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 on the PC's side.

Installing and checking the PC program

The GNS software will run on both 32 and 64 bits systems, and is compatible with both 32 and 64 bits software.

Installing the software¹

The software can be downloaded from [the GNS page of our website](#)—the software should install automatically under any “current” Windows version (from XP to Windows 11). The only requisite is having 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. Otherwise, please visit [Microsoft's .NET download page](#) 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):

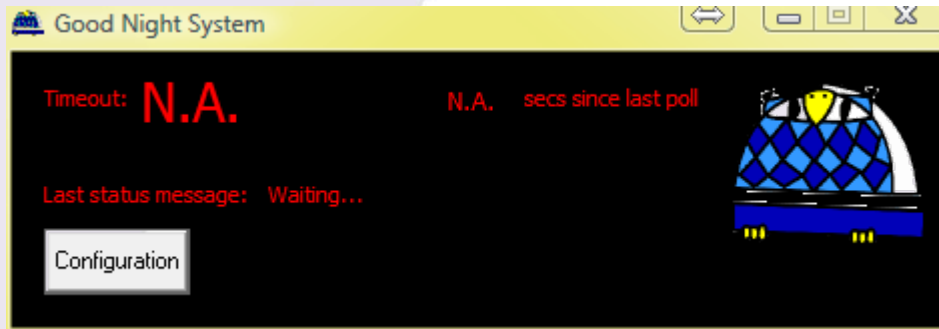
- update.vbs
- alarm.vbs
- switchoff.vbs
- idle.vbs

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 its own version.

—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:



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:

1. Run a command prompt:

Start menu → All programs → Accessories → Command prompt

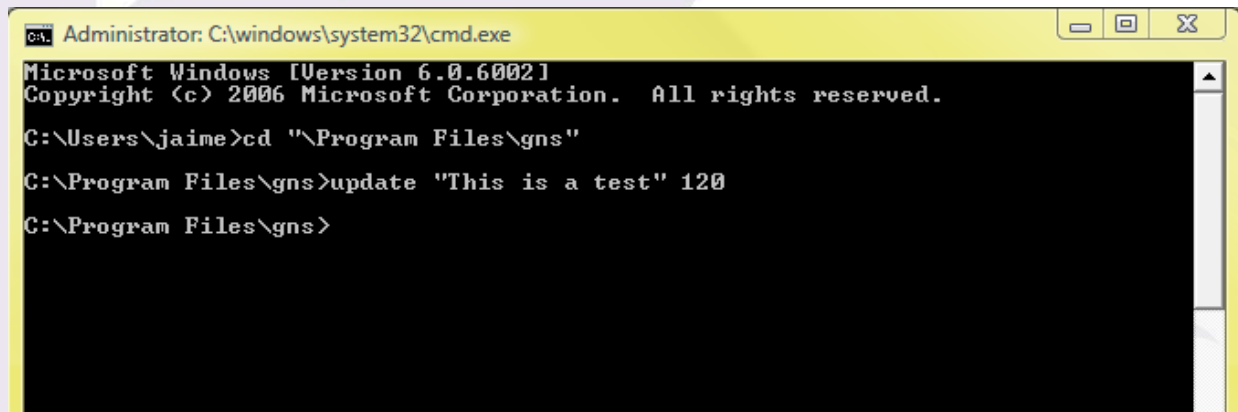
2. Go to the installation folder:

- a. For 32 bits systems, type `cd "\program files\gns"` including the quotes and followed by the return key
- b. For 64 bits systems, type `cd "\program files (x86)\gns"` including the quotes and followed by the return key²

3. Now type `update "This is a test" 120` followed by return

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

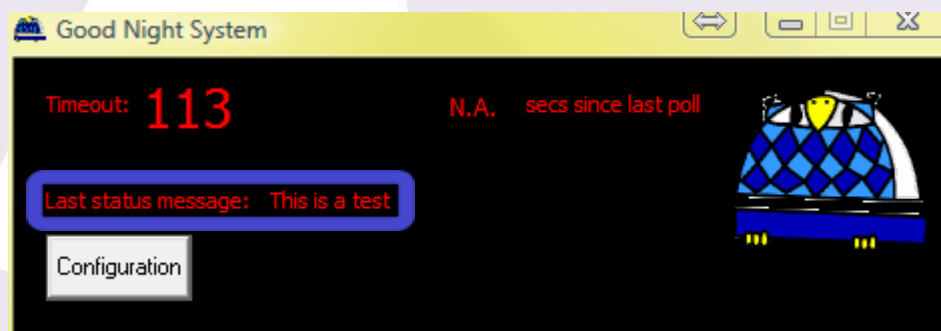
4. Your command window should look like this:



```
Administrator: C:\windows\system32\cmd.exe
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>
```

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



Note that 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 ignore the configuration button for the moment.

Let's now move on to the smartphone.

Smartphone installation

Enable your smartphone network connection (be it Wi-Fi or mobile data/3G/4G/5G, whatever).

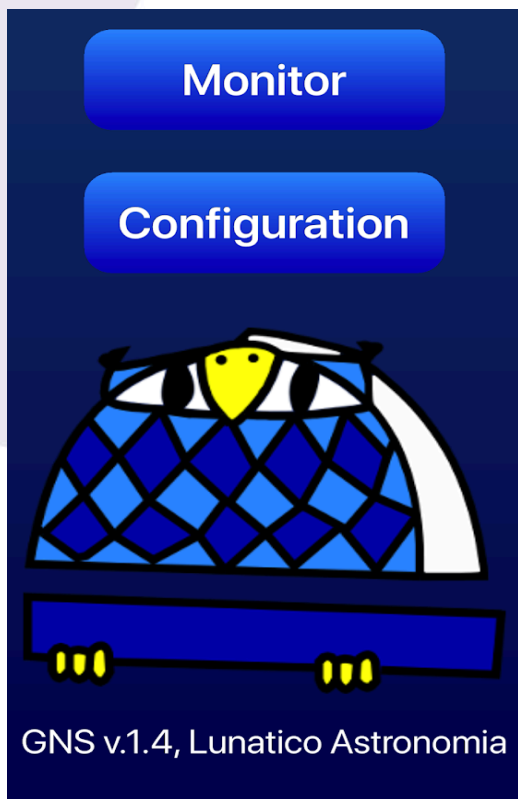
Download our Free app (from [Google play](https://play.google.com/store/apps/details?id=com.lunaticoastronomia) for Android phones, or from the AppStore, in case of iPhone—just look for Lunatico Astronomia and our apps will come up).

It is highly recommended to **try with the free version first**; it is almost identical to the full version, with only a couple of exceptions:

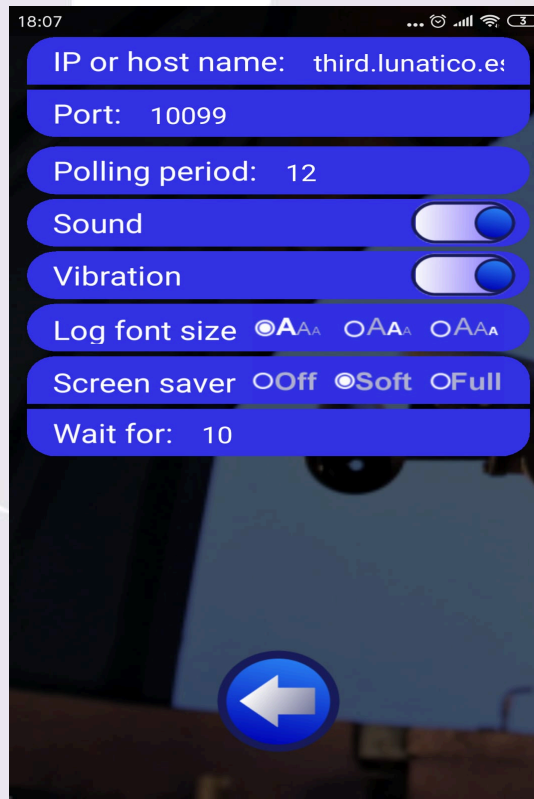
- the watchdog will be disabled and will act as a countdown timer, limiting the monitored session to 30 minutes.
- the length of the messages displayed is shorter

This way, you'll be sure everything will run smoothly before spending money on the paid version.

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



Press "Configuration" to open the parameters screen:



At this point, we have to get this app to communicate 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::18c5:4c1 :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:

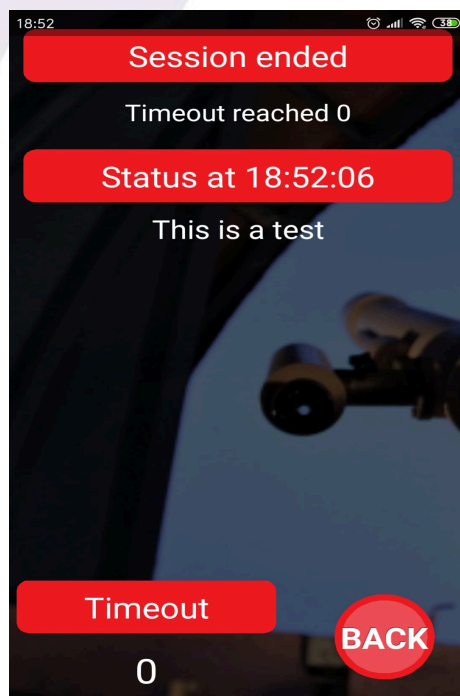
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :
```

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

- The **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).
- The **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.
- The **screen saver**, available especially for OLED displays, as they get damaged if the image is static for a long time.
 - Off—screen saver disabled.
 - Soft—display will be enabled after a message is received.
 - Full—display 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 set up, 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, when testing the PC application), the time-out reached 0 long ago.



If you got an error...

What if the communications don't go UP and you get some 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 from accepting 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:

- ★ your observatory is at home, or it's a **backyard observatory, and it shares the same network as 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 its 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 left 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 session (the address has an expiry time), and this can ruin our purpose. Please **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.

- ★ you have a **remote observatory**, connecting to it via the 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).

- ★ your **observatory is at home, but on a different network than your mobile**. This is the same, from our point of view, as the previous case.

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

If your case is either of these two last ones, please type your complete host name (similar to "myobservatory.dyndns.org") in the "Host" field of the smartphone configuration.

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 carry out, 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 website—which actually reflects the approach I'm currently using, and works great—I just update the GNS:

- once **at the start of the session**, with a 10-minute time-out—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
- and lastly, once **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 time I'm writing this, the GNS can be used with 7 popular session control systems:

Supporting the system directly (so no work needed from your part) are

- [CCDAutoPilot](#)
- [Sequence Generator Pro](#)
- [Maxpilote](#)
- [Astro Photography Tool](#)
- [ACP](#)
- [N.I.N.A.](#)

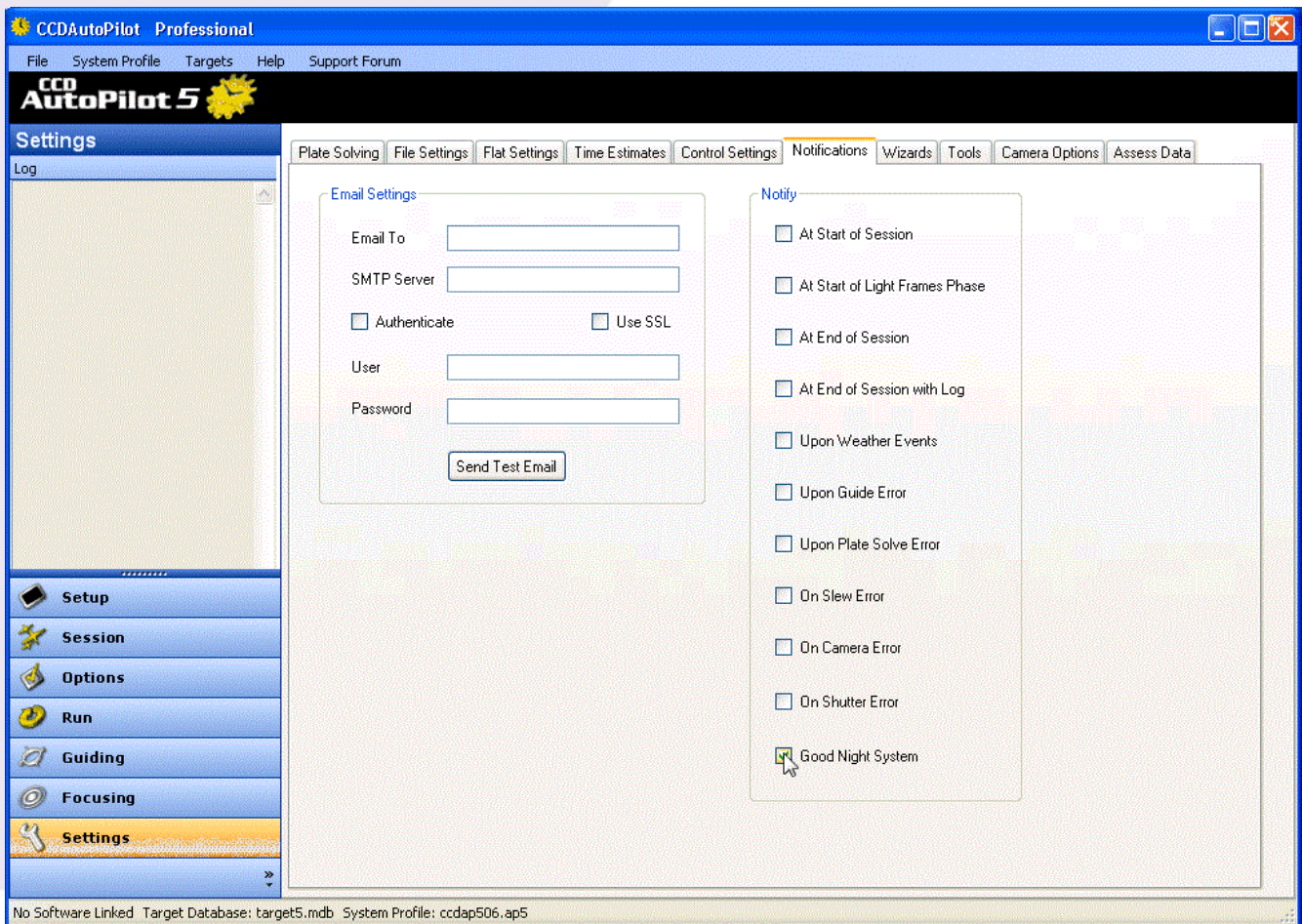
Just take a look at next section to see any installation / configuration notes (you can click on any of the names to be directed to their section).

And these can be used quite easily too, but require you to use their scripting features (please see instructions below)

- [CCDCommander](#)

CCDAutoPilot 5

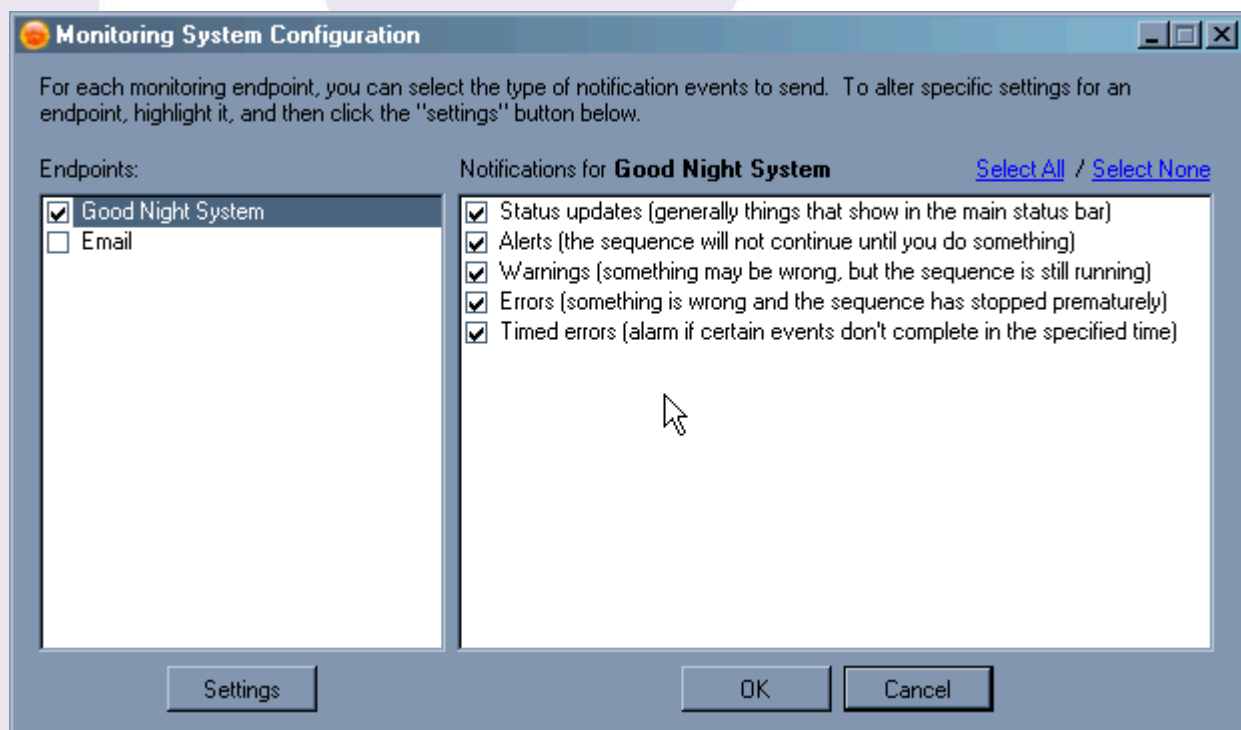
As already mentioned, CCDAP v5 includes native support for the GNS, so you'll just have to activate it from the Settings window, Notifications tab:



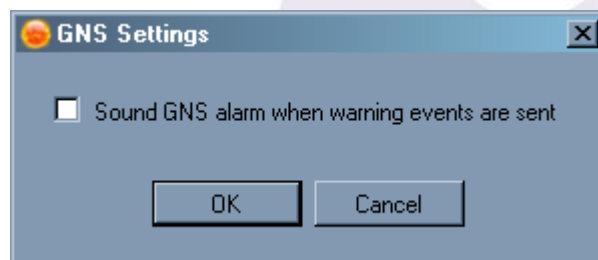
And that's all, the GNS will be launched by CCDAP and your smartphone will track your session!

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):



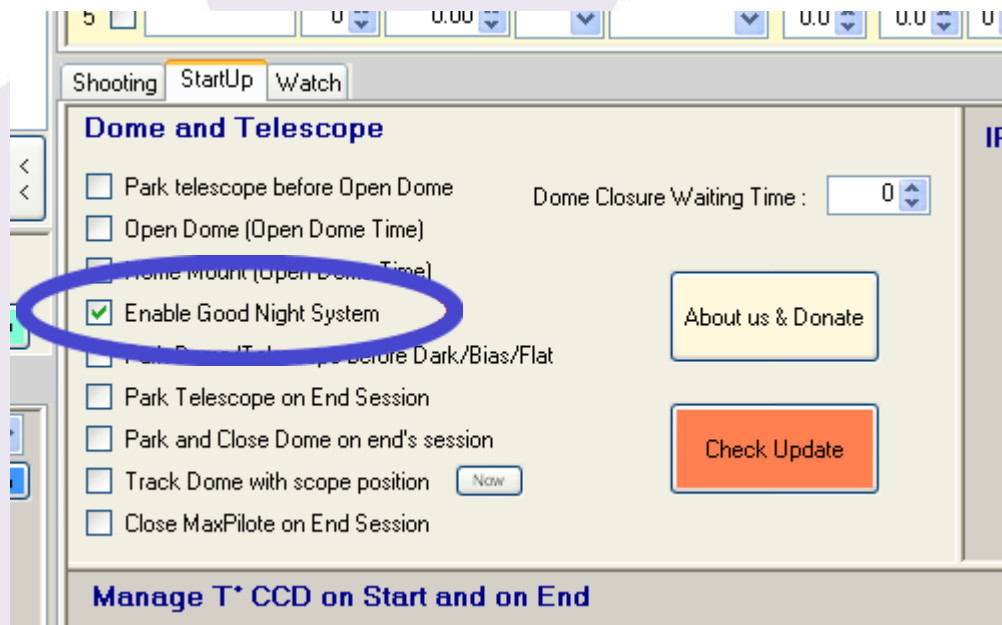
There's even an additional setting to locally play a sound if a warning is sent to the GNS:



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.

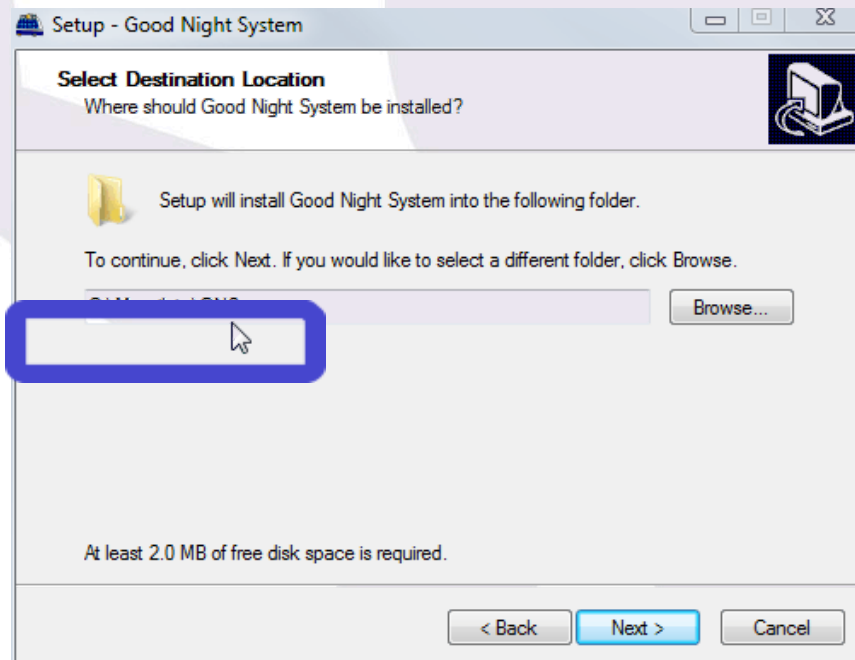
Maxpilote

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



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

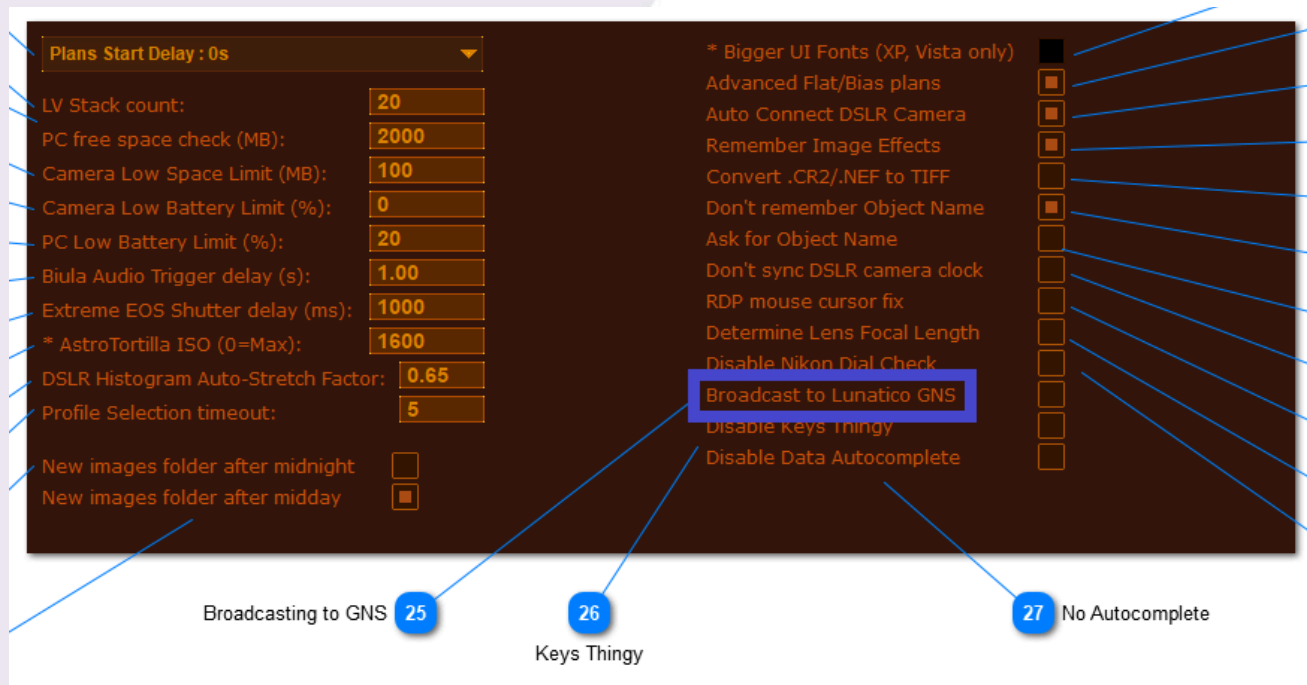
Important: for the GNS to work with Maxpilote, **the software must be installed in the same folder as Maxpilote**. Please check, as the installation program will try to install it under "Maxpilote\GNS" instead.



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

APT (Astro Photography Tool)

APT also offers GNS integration natively:

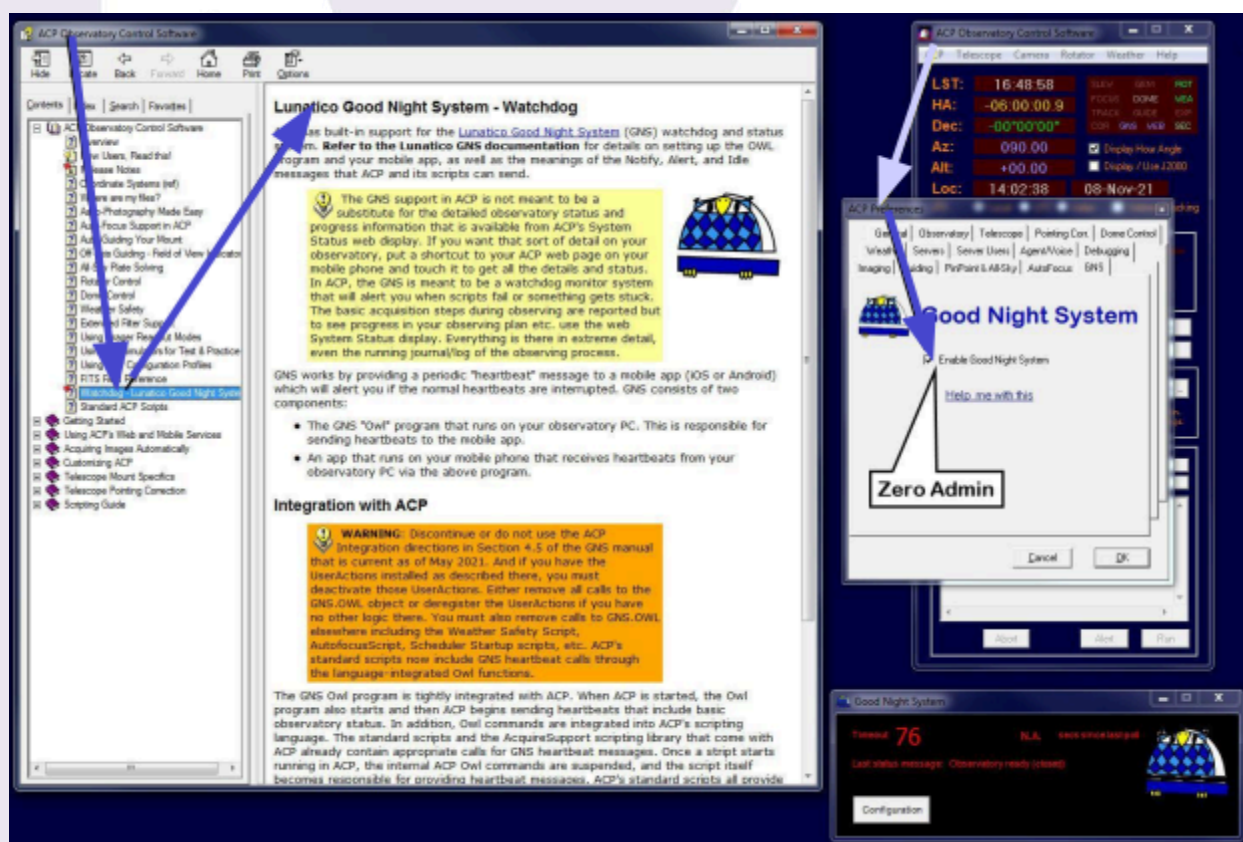


Setting number 25 allows you to activate GNS notifications. Couldn't be easier :)

ACP Observatory Control Software

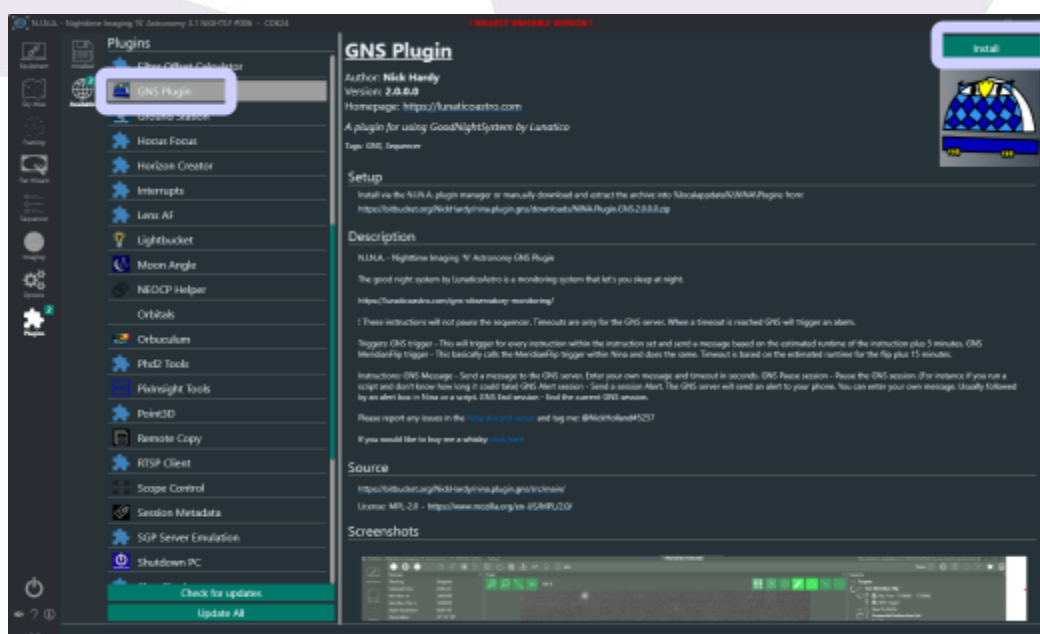
From ACP Expert version 9, **the integration of GNS has been simplified to a single click** on the GNS Preference Tab (see image below).

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.



N.I.N.A

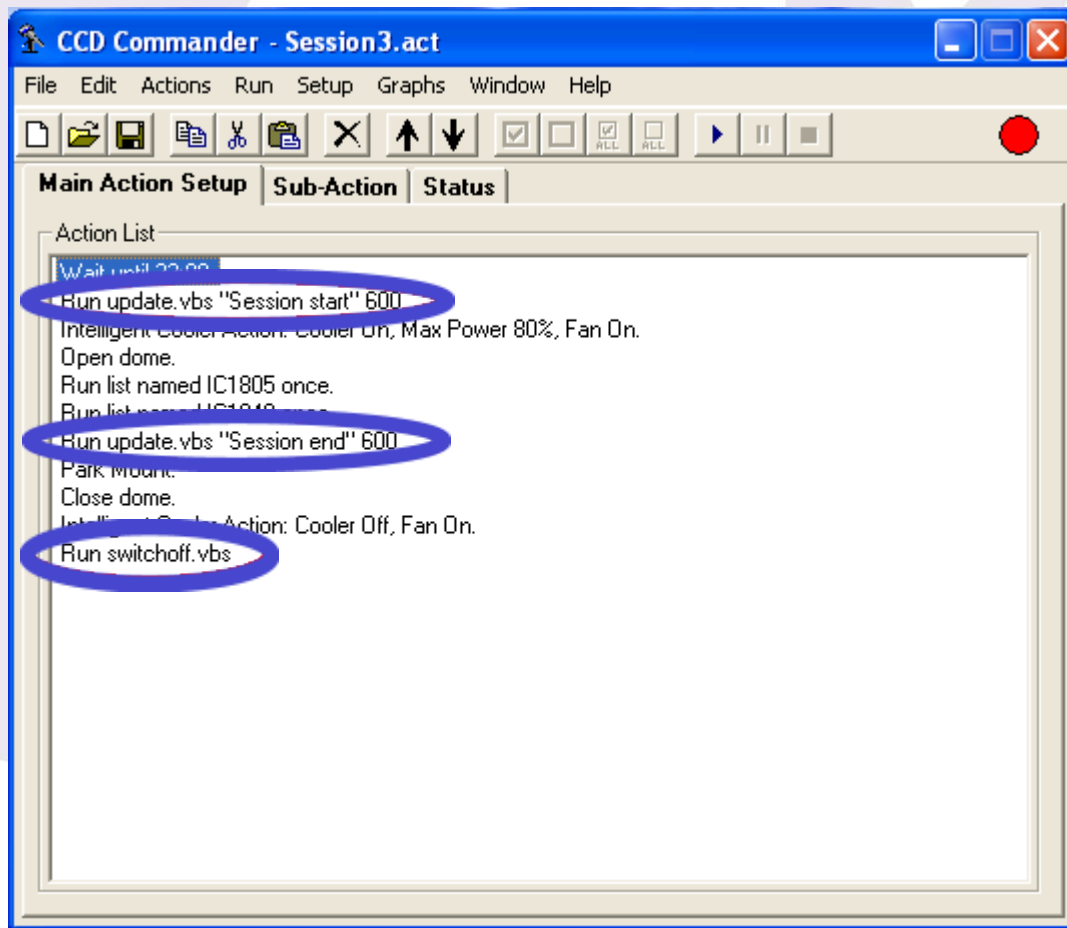
Nick Hardy has developed a GNS plugin for N.I.N.A. Enabling it is as simple as opening N.I.N.A, going to the plugins tab on the left and finding the GNS one, clicking install at the top, and then restarting N.I.N.A.



CCDCommander

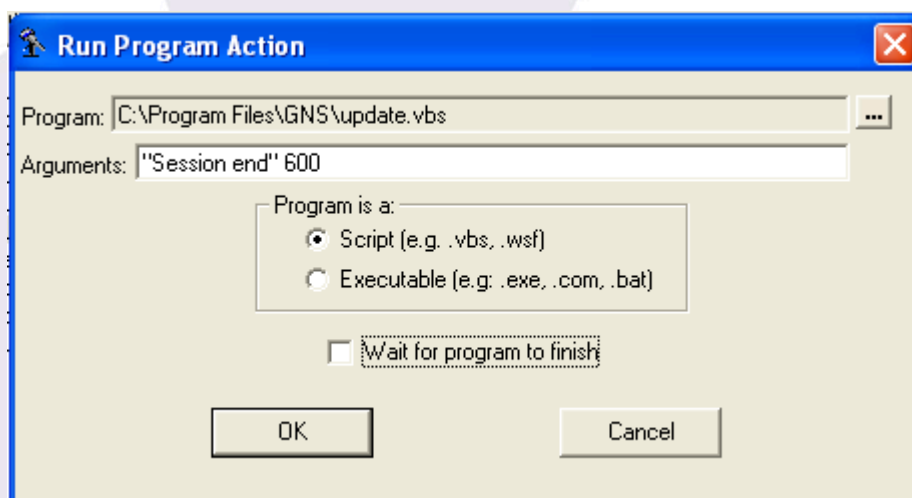
Integrating this system with CCDCommander is really straightforward, as CCDC allows the user to call scripts in many cases and at any point of the action list.

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:



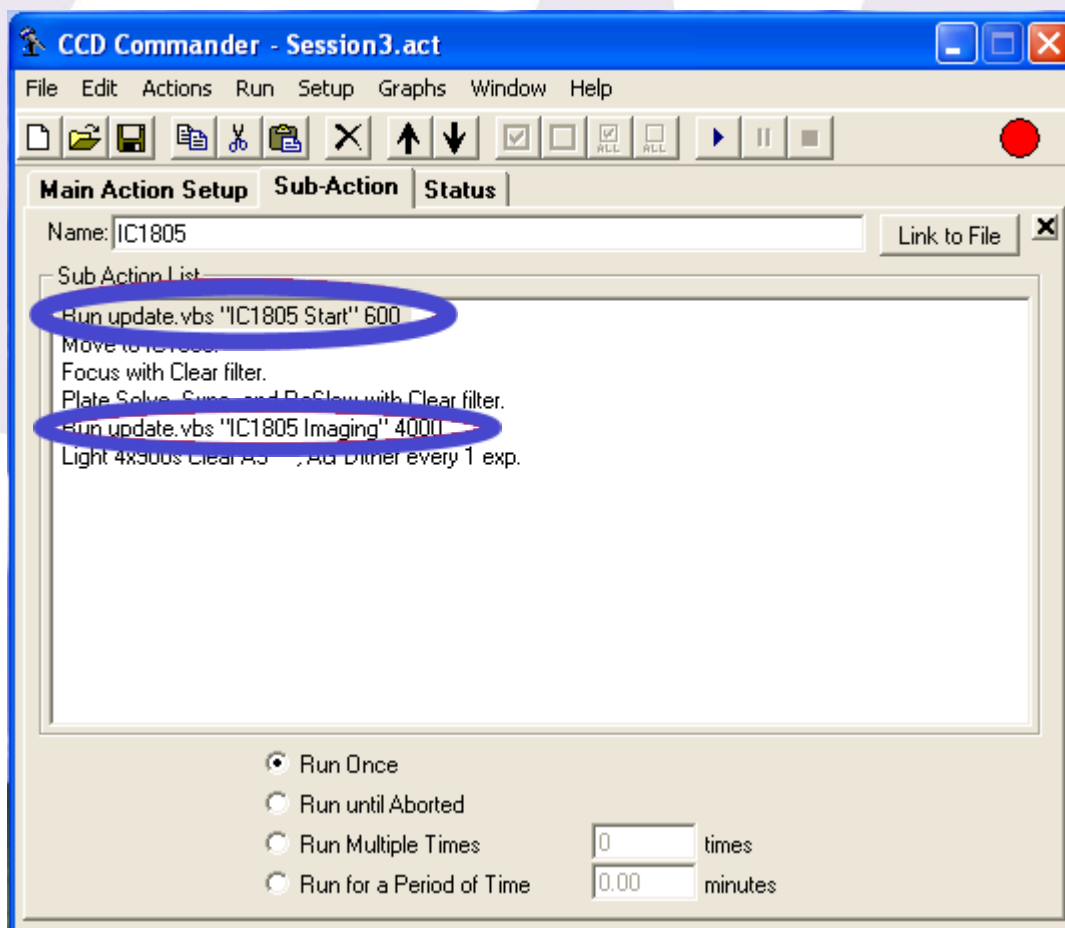
As can be seen, in the “Main Action Setup” I just added two calls to “Update”, one at the beginning of the session, another when we start winding up, and a final one to disable the system (“Switchoff”).

For each call you’ll use the “Run external program” option from the “Actions” menu:



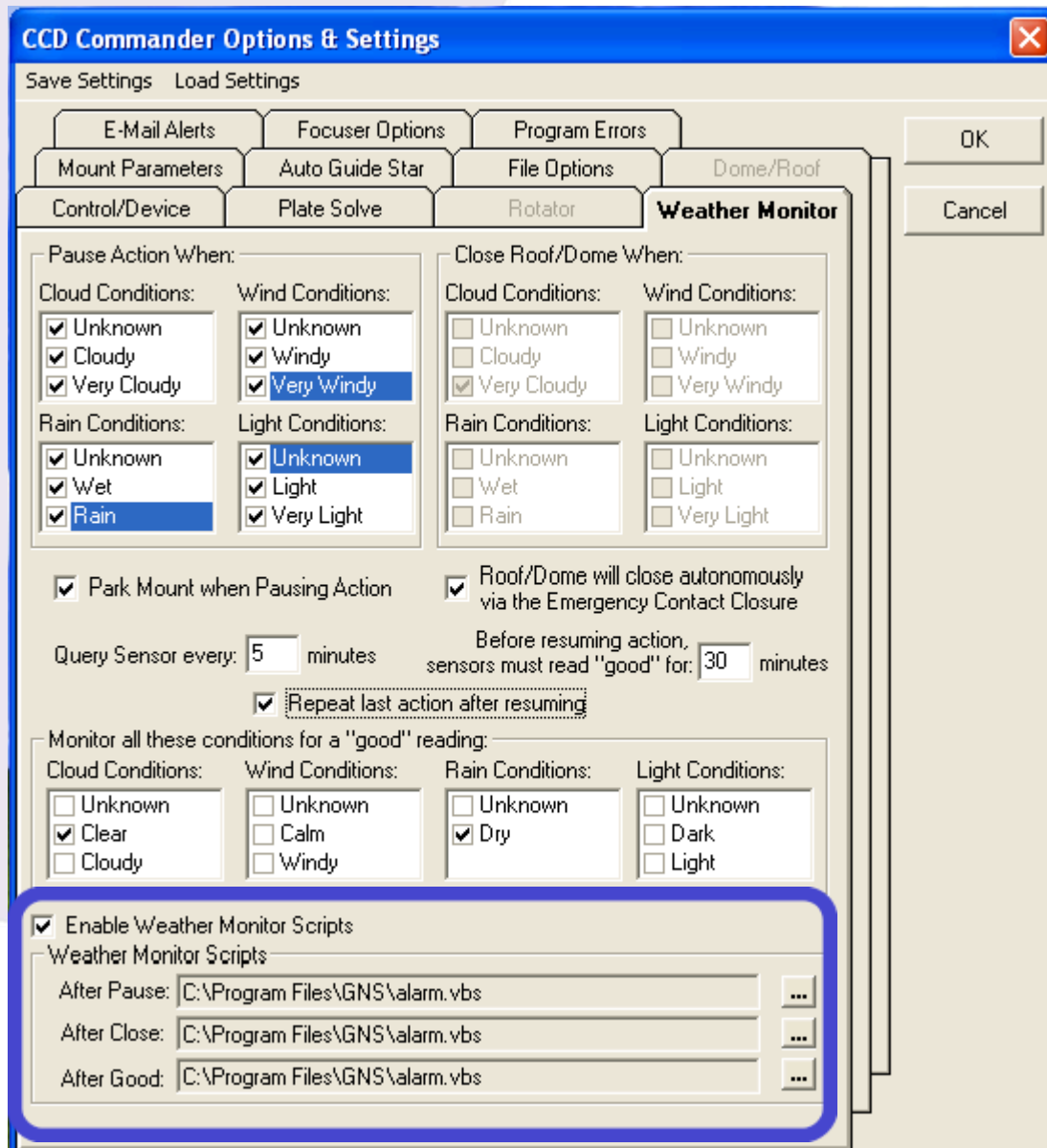
If the message includes more than one word, please enclose it in quotes (like with “Session end” in the screenshot). Also, please note we've selected “Program is a... script”, and unchecked (not really important) the optional “Wait for program to finish”.

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



Don't forget to end the main action list calling the script “switchoff.vbs”, which 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).



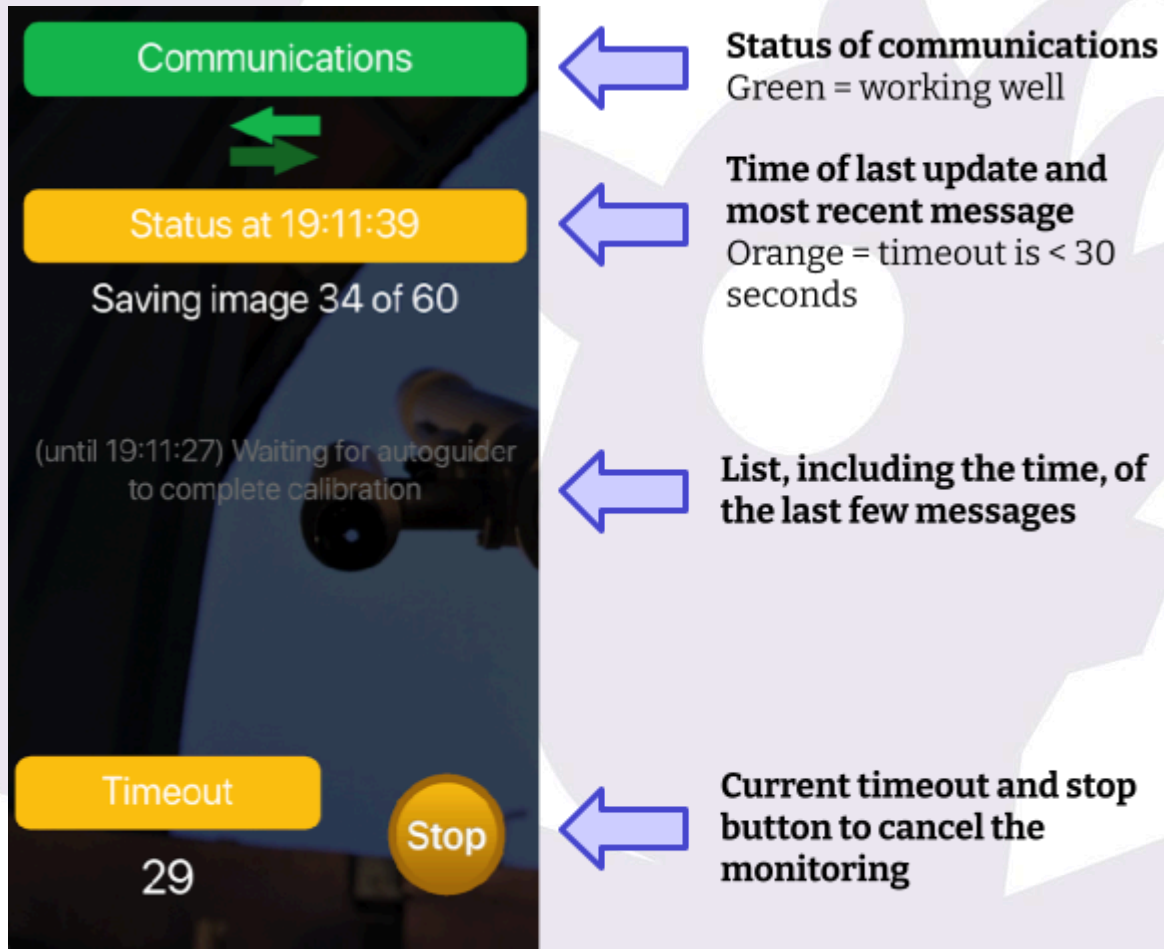
Important smartphone usage notes

Please note these few details:

—leave your smartphone fully charged or plugged to a suitable power supply – **GNS won't warn you if the battery runs out!**

- dim the screen to save battery (the screen saver is here to help)
- check your sound settings if you are using the audible alarm, and note it's associated to the media volume, not the notification volume
- 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.

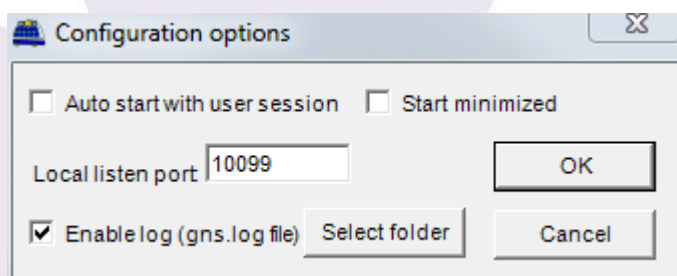
In the following image you can find an explanation of everything on the monitor screen.



The time-out shown in the smartphone will typically be delayed by a few seconds compared to the one in the PC; this is normal.

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**: if checked, the GNS program will be launched and will start accepting connections as soon as you log in on Windows. Can 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 every time it is launched, to avoid cluttering your desktop. 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; there is 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**: this will create—or append to if it already exists—a text file describing the session events, from the point of view of the Windows software. This feature is useful for troubleshooting. 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.

And on a sidenote, **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.

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 time-out (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 time-out is reached. And 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, they are 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 installation 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 time-out:
 - if timeout > 0, the behaviour 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 time-out according to 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](#).

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 session → GNS close session

The key points are:

- The time-outs can be generous, no need to risk a false alarm. We just want to detect if something happens (to your software, to Windows—more likely—, ...) 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, with no pauses. While paused, its protection is effectively disabled. It would be great to avoid this.
- The status message can be informative, but if it's too long, it 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 on every phone.

I hope this explanation is detailed enough, but of course, I'll be happy to help if any questions come up, just reach out at support@lunaticastro.com.

Pending features and wish-list

As of today, the “wish-list” has been reduced to a 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.

Last notes

Warranty (or lack thereof!)

I'm confident you understand there's no warranty, whatsoever. Use the software at your own risk. In the unfortunate scenario that your expensive equipment gets damaged for any reason, we won't take any blame for it.

Acknowledgments

Developing the system has taken much longer than expected, for a variety of reasons, but it's been worth the effort. I'd like to take a moment to acknowledge a few people who have helped me throughout.

- 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.

–1.0.1.2 Routine text revision and format update. N.I.N.A integration section added.
