



NightFront for N.I.N.A.

Plugin Guide

Version 3.19.1.0

Contents

NightFront (NINA Plugin)

- What the plugin does
- Plugin requirements
- Installing the plugin
- Plugin quick start
- Plugin documentation
- Questions and problems

Sequence instructions and triggers

- A recommended sequence skeleton
- Nightly Update instruction
- NightFront Container sequence item
- Replan instruction
- Seeing Trigger
- Wind Speed Trigger
- Before Target and After Target triggers
- While Work Remains loop condition
- NightFront Filter Offsets instruction
- NightFront Save Night's Log instruction

Example sequence and templates

- Using them
- The example night
- The five templates
- Why it is laid out this way

The Options tab

- Options tab Setup page
- Options tab Targets page
- The plugin's collected-frames ledger

Filter offsets

- Kept up to date, not calibrated once
- How a measurement works
- When an offset is written
- The NightFront Filter Offsets instruction
- The Filter offsets page
- Stopping or undoing
- Set up NINA once, at the end
- A recommended arrangement
- If you already use filter offsets

Calibration flats in NINA

How flat requirements accumulate

Shooting the flats

Rotating without shooting

Loop conditions for flats

Example flats sequence

When there's nothing left to shoot

Flat settings on the Options tab

NightFront (NINA Plugin)

Non-Dominate the Night

NightFront orchestrates your astrophotography over a whole season, for all of your targets at once: the desktop app finds and frames targets, chooses the months to image them, plans each night for every project in rotation, and processes the frames until the data says each project is done.

This plugin is the business end, where the imaging actually happens. It runs on your imaging computer, inside NINA, unattended, and drives every night toward the two goals you set in the app: **utilization**, getting as much of the dark as possible onto the sensor, and **quality**, getting it under the best sky available. It is where the plan becomes real.

Each night is planned from your own per-filter **exposure plans** (sub length, gain, offset, how many frames per filter) and per-filter **moon avoidance profiles**, enforced as a hard constraint. The planner solves the whole night as one problem, against:

- the **forecast** for your site (cloud cover, transparency and seeing), from the free 7Timer! service by default or from Astrospheric if you have an API key;
- **live readings from your ASCOM weather device** as the night runs, which can trigger a replan of the rest of the night when the sky genuinely turns;
- your rig's **real overheads**: slews, meridian flips and dithers all cost time in the plan;
- and what each project still needs.

The planner ships **inside this plugin** as `nightfront-cli.exe`, a build of the NightFront desktop app's scheduler, installed beside the plugin. There's nothing extra to install on the imaging computer. Everything upstream (your targets, the balance of quality against utilization, your filter rules, your rig and the season's project queue) is set in the desktop app, and the frames the plugin collects flow back into the app's processing workflow; the NightFront User Guide covers all of that.

What the plugin does

- **Plans the night at dusk. Nightly Update** imports the plan you exported from the desktop app, or, if there isn't one for today, generates one on the spot from your NightFront config and the live sky.
- **Runs it as ordinary NINA instructions.** The **NightFront Container** fills with slewing and centering, guiding, filter changes, exposures and dithers, and NINA executes them like a sequence you built by hand. Meridian flips and refocusing are left to your own NINA triggers; the plan budgets time for flips but adds no flip or autofocus instructions.
- **Answers to the sky. Replan** re-solves the rest of the night after a safety interruption or when a **Seeing Trigger** or **Wind Speed Trigger** fires; gusty hours can switch to shorter subs without giving up the night's integration.
- **Remembers progress.** The plugin counts the frames it shoots and feeds them back to the planner, so a backlog survives several unattended nights before you sit down to review.

- **Knows when to stop. While Work Remains** stops an unattended observatory opening once every Active target is done.
- **Sets up your flats.** It records the filter, measured rotator angle, gain, offset and readout mode of every exposure, and a set of flats instructions shoots matching flats later.
- **Works on the imaging computer.** The Options tab's Targets page shows what every target still needs, lets you change requested frame counts, checks what tonight's sky allows, and creates targets from NINA's Framing Assistant — or re-points an existing one onto what you have framed. ⚠️ A re-point discards that target's accepted frames, since they no longer describe where it is looking; the page says so and asks first.
- **Uses NINA's custom horizon.** If your NINA profile names a horizon file, the planner uses it for any config that doesn't carry its own.

Plugin requirements

- **Windows** with **N.I.N.A. 3.0** or later.
- **A NightFront config:** the `.json` the NightFront desktop app saves (point the Options tab at it), or the `session-config.json` an export from the desktop app writes into your data folder. Build it in the desktop app; the imaging computer doesn't need the app installed, just the file.
- **A folder shared with the desktop computer.** When the desktop app runs on another machine, the config and the plugin's data folder must live in a folder both computers share and keep in sync (Google Drive, OneDrive, Syncthing, a network share or similar), so the plugin plans from the file you last saved and finds the plans you export. Save in the app well before dusk so the sync has finished.
- **Internet access for generated plans and replans,** to fetch the forecast. The default source, 7Timer!, is free and needs no key. An Astrospheric API key in your config is optional.
- **Optional:** an ASCOM weather device for live conditions and the Seeing and Wind Speed triggers (the Seeing Trigger needs one that reports star FWHM), and a rotator for angle-matched flats.

Installing the plugin

1. **Close NINA.** The installer replaces the plugin's files, and a running NINA holds them open.
2. **Run the NightFront NINA Plugin installer** (`.msi`). It installs for your Windows user only, so it doesn't ask for administrator rights, and it puts the plugin where NINA looks for it:
`%LOCALAPPDATA%\NINA\Plugins\3.0.0\JeffRidder.NINA.Nightfront.`
3. **Start NINA.** The NightFront items appear in the sequencer under the **NightFront** category, and the settings under **Options** → **Plugins** → **NightFront**.

Installing a newer version replaces the old one, clearing out files earlier versions left behind. The plugin isn't listed in NINA's built-in plugin browser.

The installer also puts a **NightFront NINA Templates** folder on your Desktop: an annotated example sequence and five templates showing where each NightFront item goes. The folder is replaced on every upgrade, so copy a file before you edit it. The *Example sequence and templates* page explains them.

This guide is installed with the plugin as a PDF. To open it, press **Plugin Guide** at the top of the **Setup** page under **Options → Plugins → NightFront**.

Plugin quick start

1. **Options → Plugins → NightFront → Setup:** set the **NightFront Config (for online planning)** to the `.json` the desktop app saves, in the folder this computer shares with the desktop. Leave the **NightFront Data Folder** clear: the plugin then keeps its files in the config's folder, which is also where the desktop app exports plans.
2. **Build a sequence** like this, with your own startup, safety and shutdown steps around it:

```
"Every night" (loop)          ← condition: While Work Remains
├── "Start of night"
│   ├── Wait for Time: sunset
│   ├── Unpark Scope
│   └── Nightly Update
├── "Imaging until dawn" (loop) ← condition: Loop until nautical dawn
│   ├── "Imaging" (loop)      ← condition: Loop while safe
│   │   ├── triggers: Meridian Flip, Center After Drift, Before Target,
│   │   │           After Target, Seeing Trigger, Wind Speed Trigger
│   │   └── NightFront Container
│   └── "Safety recovery": park, wait while unsafe, unpark, then Replan
├── "Sky flats" (loop)
├── "End of night": park, NightFront Save Night's Log
└── Wait for Time: sunrise, offset +5 minutes
```

The Desktop folder **NightFront NINA Templates** holds a complete, annotated sequence along these lines to start from; [Sequence instructions and triggers](#) says why each part sits where it does.

3. **Check it on the Targets page:** press **Reload** to see what each target still needs, and **Check tonight** to see what tonight's sky allows.
4. **Run the sequence.** Nightly Update's status line tells you whether it imported a plan file or generated one, and the NightFront Container shows each target's progress as NINA works.

Plugin documentation

- [Sequence instructions and triggers](#): Nightly Update, the NightFront Container (and how it fills the rest of a night whose plan finishes early), Replan, the Seeing, Wind Speed, Before Target and After Target triggers, While Work Remains, Save Night's Log, and a recommended sequence.
- [Example sequence and templates](#): the annotated example night and the five templates the installer puts on your Desktop (start of night, imaging, safety recovery, sky flats, end of night), how to load them, and why they are laid out as they are.
- [The Options tab](#): the Setup page (data folder, config, Planning Mode, Plan Selection, filling the rest of the night), the Targets page (Active and Planned columns, moving and enabling/disabling targets, editing requested counts, Check tonight, framing a target or a whole mosaic in NINA's Framing Assistant, creating targets from it and re-pointing existing ones), and the collected-frames ledger.

- **Filter offsets:** offsets kept up to date every clear twilight rather than calibrated once, and why; what is tested (parfocality included) before a value is written; the NightFront Filter Offsets instruction; the Options tab's Filter offsets page with its history and Restore; stopping or undoing at any time; and the NINA settings to change once.
- **Calibration flats:** how flat requirements accumulate, the flats instructions and loop conditions, and an example flats sequence.

For everything on the desktop side (building a config, the objectives, filter rules, priority order, the planners and the season-scale roadmap), see the NightFront User Guide.

Questions and problems

Ask on NightFront's Discord server: <https://discord.gg/65ZYummSwb>. The desktop app opens it from the **Community** button at the foot of its left-hand rail.

For anything the plugin did — Nightly Update not importing or generating a plan, Replan doing nothing, a trigger that never fired — bring **NINA's own log** from the night in question (%localappdata%\NINA\Logs), and say which versions of the plugin and the desktop app you are running. The two ship separately and are allowed to differ, so the pair is often the answer.

Sequence instructions and triggers

Everything the NightFront plugin adds to NINA's Advanced Sequencer lives under the **NightFront** category in the palette. This page covers the items you use to plan and run a night. The calibration-flats items have their own page, [Calibration flats](#), and the plugin-wide settings are on [The Options tab](#).

Palette name	Kind	What it's for
Nightly Update	Instruction	Loads tonight's plan into the NightFront Container, importing or generating it
NightFront Container	Container	Holds tonight's imported targets and shows their live progress
Replan	Instruction	Re-solves the rest of the night from what has actually been shot
Seeing Trigger	Trigger	Runs your instructions when star FWHM crosses a threshold
Wind Speed Trigger	Trigger	Runs one set of instructions as wind rises, another as it drops
Before Target	Trigger	Runs your instructions just before each NightFront target starts
After Target	Trigger	Runs your instructions right after each NightFront target ends
While Work Remains	Loop condition	Keeps an unattended sequence looping only while targets still need time
NightFront Filter Offsets	Instruction	Measures each filter's focus offset by bracketed autofocus and writes it to NINA's offset table
NightFront Save Night's Log	Instruction	Saves the night's NINA log beside your NightFront config, so the app can analyze the whole night

A recommended sequence skeleton

A typical unattended sequence looks like this. It is the shape of the example night the installer puts on your Desktop (below). Names in quotes are containers you create yourself; everything in **bold** comes from the plugin.

```

Sequence
├── "Every night" (loop) ← condition: While Work Remains
│   ├── "Start of night"
│   │   ├── Wait for Time: sunset
│   │   ├── ...your own startup: connect, cool the camera, open the roof...
│   │   ├── Wait for Time: nautical dusk
│   │   ├── ...your own first focus: Unpark Scope, slew to a star field, autofocus...
│   │   ├── NightFront Filter Offsets (optional)
│   │   └── Nightly Update
│   ├── "Imaging until dawn" (loop) ← condition: Loop until nautical dawn
│   │   ├── "Imaging" (loop) ← condition: Loop while safe
│   │   │   ├── triggers: Meridian Flip and Center After Drift (NINA's),
│   │   │   │   Before Target, After Target, Seeing Trigger, Wind Speed Trigger
│   │   │   └── NightFront Container
│   │   └── "Safety recovery"
│   │       ├── ...your own weather-stop steps: Park Scope, close the roof...
│   │       ├── "Wait while unsafe" (loop) ← condition: Loop while unsafe
│   │       ├── ...your own recovery steps: reopen the roof, Unpark Scope, cool the camera...
│   │       └── Replan
│   ├── "Sky flats" (loop) ← conditions: NightFront While Flats Remain,
│   │   Loop while safe, Loop until sunrise +30 min
│   ├── "End of night"
│   │   ├── Park Scope
│   │   ├── ...your own shutdown: close the roof, warm the camera...
│   │   └── NightFront Save Night's Log (last)
│   └── Wait for Time: sunrise, offset +5 minutes
└── End area: anything for when the whole sequence stops

```

The rules that matter:

- **Nightly Update runs once, near the start of the night, before the NightFront Container.** The container can sit later in the same branch or be nested inside a later container, such as the loop that repeats until dawn.
- **Anything you run every night before imaging goes before Nightly Update,** most of all NightFront Filter Offsets. A plan Nightly Update generates starts at the moment it runs, so it then plans only the time that is actually left; placed earlier, it plans targets into time something else spends, and they are cut short or skipped ([Filter offsets](#)).
- **Imaging and its safety recovery sit together inside the loop that runs until dawn,** recovery after imaging. As separate branches of the night, the first weather stop would fall through to the end of the night, and flats, park and Save Night's Log would run in the middle of it.
- **Every Park Scope needs an Unpark Scope after it, before the next imaging.** A parked mount refuses every slew, so a weather stop that parks must unpark in the safety recovery, after the wait and before Replan, and the end of the night's park must be undone in the next night's first focus. The example night does both. A rig whose roof must close over a parked mount reopens the roof before Unpark Scope.
- **Replan can go anywhere in the same sequence.** It finds the container on its own. The natural places are the safety recovery and the drop areas of the weather triggers.
- **Before Target and After Target must sit on the NightFront Container or one of its ancestors** (for example, on "Imaging"). A sibling branch such as "Safety recovery" never sees the container's targets run.
- **Each night opens with a wait for sunset and closes with a wait for a few minutes past sunrise,** so each pass of "Every night" starts in the evening. The closing wait needs its offset ([Example sequence and templates](#)).

- **Dawn flats need a time limit.** After sunrise a sky flat fails and stays needed, so a flats loop with no Loop until condition would retry it all day.
- **Your own NINA triggers do the refocusing and the meridian flips.** The plan budgets time for a flip, but the plugin adds no autofocus or meridian-flip instructions of its own.

The installer puts a complete, annotated version on your Desktop, built only from NINA's own items and NightFront's: an example night and five templates, in the **NightFront NINA Templates** folder. Stock NINA has no "once safe" instruction, so there the wait for safe weather is built from NINA's own **Loop while unsafe**. See [Example sequence and templates](#).

Nightly Update instruction

Gets tonight's plan into the NightFront Container, whichever way it has to. When it runs, it looks in your NightFront data folder and takes the first of these that applies:

1. **A plan file for today.** A plan the desktop app exported, named `TargetsForTonight_<date>.json` for today's date (for example `TargetsForTonight_2026-09-14.json`), is imported as-is. The name has to match exactly — any other `.json` file in the folder is left alone, whatever its name, so a config, a backup or a file your cloud-sync client leaves behind is never mistaken for a plan.
2. **A plan it generated earlier tonight.** If a sequence restarts after a safety interruption, Nightly Update re-imports the `NightFrontPlan.json` it already generated tonight instead of solving again. Replan is the way to re-solve on purpose.
3. **Otherwise, it generates a plan.** It reads your NightFront config (see [The Options tab](#)), captures live conditions from NINA's weather device, and runs the planner that ships inside the plugin. The plan starts at the moment it runs, or at the start of the config's planning window if that is later.

Either way, the container fills with ordinary NINA instructions: slew, center and rotate, start guiding, filter changes, exposures and dithers. NINA runs them exactly as it would a sequence you built by hand.

Nightly Update fields and status

On screen	What it means
If generating, plan for up to N s	The Time Budget: how long the optimizer may search before it uses the best schedule found so far. Default 180 s. Ignored when a plan file is found.
Status dot and message	Green after a successful import or generate, red after a failure. The message names the plan file, or reads like <code>Planned at 20:41 (4 target(s))</code> .
Last checked:	When it last ran.
Amber advisory line	Appears when your NightFront config was last saved 7 or more days ago.

- **The Time Budget is wall-clock time**, so it means the same thing on a mini-PC as on a workstation; a slower machine simply searches less far in the same time. It saves with the sequence, so two sequences can use different budgets.
- **Under Greedy or Proportional planning the Time Budget does nothing.** Both are a single, instant pass. Planning Mode is set on the Options tab; see [Planning Mode setting](#).

- **When the forecast came from anywhere other than Astrospheric**, the status line names the source, for example `- forecast: 7timer` (7Timer is the free default). If the plan rests on a forecast that couldn't be refreshed and was served from an expired cache, the line reads `- forecast: stale-cache` and you get a warning notification rather than a plain success.

When Nightly Update produces no plan

Status message	What happened to the container
All targets already complete - nothing to plan	Emptied, so the night doesn't re-shoot finished work
No astronomical darkness remains tonight	Left as it was
No forecast available and no cached one to fall back on	Left as it was
No NightFront config found	Left as it was
Another NightFront CLI run is already in progress	Left as it was

Note: A forecast that can't be loaded is reported, never invented. NightFront uses a live forecast, then a cached one, then a stale cached one up to 48 hours old, and only then gives up. Giving up leaves any plan already loaded in place.

Heads up: The config-age advisory is only advice. An old config never stops the night: the plugin's own collected-frames ledger carries the frames shot since your last save, so an old file can only overestimate what is still needed. Saving the config in the desktop app makes the counts exact again.

What NINA flags before the night starts

Nightly Update reports a validation issue only when no plan can possibly be produced:

- neither the NightFront data folder nor a NightFront config is set (a clear data folder uses the config's folder),
- no NightFront Container can be found in the sequence, or
- there is no plan file for today **and** either the planner or a NightFront config is missing.

NightFront Container sequence item

Holds the instructions Nightly Update or Replan imports. You add it empty; the plugin fills it.

Inside the sequencer it shows `N target(s) scheduled` and one summary line per target:

Column	Shows
Dot	The target's run status
Name	The target (hover for the full name, useful for long mosaic panel names)
Coordinates	RA and Dec
Rot :	The planned rotation angle
Window	The target's slot in tonight's plan
N/M subs	Exposures completed so far out of those planned, updated live as NINA works

When the plan finishes early

A plan can run out before the night does: NINA often works faster than the plan allowed for, or the last target simply finishes ahead of dawn. When that happens and usable darkness is left, the NightFront Container **replans the rest of the night by itself and keeps imaging**, without a Replan instruction or any other template step. The container doesn't report finished until the refilled plan has run.

- **Usable darkness** means at least 30 minutes, or the longest minimum imaging time among your enabled Active targets plus one slew, whichever is more. With the usual 60-minute minimum and a 5-minute slew allowance that is 65 minutes. The night's end is dawn as the plan itself has it: the end of the last target's time window.
- **If targets still owe time**, the refill is an ordinary replan of what remains.
- **If every enabled Active target is already complete**, the refill keeps imaging the best-placed of them with extra subs. Requested is exceeded on purpose; nothing is written to your config. Planned, disabled and completed targets are never used.
- **It uses Replan's machinery**: the same planner, the same 5-minute limit, the same collected-frames ledger and the same plan file, so a later Replan or the next Nightly Update sees what the refill installed. Before Target and After Target fire for refilled targets as usual.
- **A hand-built or generated night is not refilled**, unless a Replan instruction in the same sequence has **Replan hand-built plans** ticked. See [Replan and hand-built plans](#).
- **At most 3 refills a night**, never two within 5 minutes, and a refill whose plan took no exposure stops refilling for the night. A refill that fails (no forecast, the planner busy or missing) ends the container normally, with one log line and one notification, and isn't retried that night.
- **It never refills when the sequence stopped the plan** (a loop condition or the safety condition ended it, or you stopped the sequence), or while your safety monitor reports unsafe. Recovery is your safety recovery's job.

The container shows what happened on a line under the target count, for example:

Line	Meaning
Plan finished at 03:12 with 2 h 10 m of darkness left – replanning to fill it	A refill is running
... – replanned to fill it: M 31, M 33	Targets still needing time, and the refill planned them
... – filling with extra subs on M 31 (all targets complete)	Everything was complete, so M 31 gets extra subs
... – less than 1 h 5 m left, so not refilling	Not enough darkness left
... – not refilling the hand-built plan "Tuesday" (...)	A hand-built night, and no Replan opts in
... – refill limit reached (3 tonight)	Three refills already tonight
... – the refill failed (...); not refilling again tonight	The refill couldn't run

Turn it off with **Fill the rest of the night when the plan finishes early** on the Options tab's Setup page. Off, the container simply ends when its plan does, as it always did.

Center After Drift with the NightFront Container

If you use NINA's own **Center After Drift** trigger, put it on a container *above* the NightFront Container (on "Loop while safe", say). The plugin keeps it pointed at whichever target is running, updating it each time a target's centering starts.

This matters because NINA looks for that trigger's coordinates by walking *up* from where the trigger sits, and the imported targets are *below* the container. Left alone, the trigger would re-center on RA 0, Dec 0 all night.

Custom horizon from NINA's profile

When NINA's own **Options** → **Astrometry** → **Custom Horizon** points at a horizon file, the plugin sends that file to the planner for Nightly Update, Replan and **Check tonight**. It's read live from the active profile each time, so re-pointing it in NINA takes effect on the next plan.

- **A horizon saved in your NightFront config wins.** NINA's horizon only fills the gap for a config that doesn't carry one.
- **A missing or unreadable horizon file doesn't stop planning.** The planner warns and falls back to the config's horizon, then to the flat minimum altitude.
- The Options tab shows what NINA has configured; see [Custom horizon readout](#).

Replan instruction

Re-solves **the remainder of the night** and swaps the new plan into the NightFront Container. When it runs, it:

1. records what the container has actually captured so far tonight,
2. reads live conditions from NINA's weather device,
3. re-runs the optimizer on what's left, and

4. repopulates the container. The previous plan file is kept in a `replan-history` folder.

Replan always uses the optimizer and your **Plan Selection** setting, whatever **Planning Mode** is set to. It plans only enabled, Active targets that aren't marked complete.

Where to place Replan

- **In your safety recovery**, after imaging, as in the skeleton above. After clouds or wind force a stop, the night resumes from a plan that reflects the hours you lost, instead of a schedule that no longer fits.
- **Inside a trigger's drop area** (Seeing Trigger, Wind Speed Trigger), so the plan responds to the sky changing while the roof is still open.

Either place is safe. When a Replan runs from a trigger while a target is imaging, that target ends at its next step, with no exposure cut short, and the new plan takes over straight away. Your meridian flip, autofocus, guiding and weather triggers and your safety condition keep covering it the whole time. In the safety recovery the NightFront Container isn't running, so the new plan simply waits for imaging to resume.

Replan fields and status

On screen	What it does
Exposure variant	Optional. The name of an exposure variant from your NightFront config. The remaining sky time is shot at that variant's sub lengths. Leave blank, shown as <code>(none - normal exposures)</code> , for your normal exposures.
Dot and short message beside it	Whether the name matches a variant in your config (see below)
Replan hand-built plans	Off by default. See Replan and hand-built plans .
Status dot and message, Last replanned:	The outcome of the last run, for example <code>Replanned at 01:12 (3 target(s))</code>

The exposure variant list comes from the same NightFront config Replan plans from: the config set on the Options tab, or the exported `session-config.json` if none is set. You can also type a name.

Dot	Message	Meaning
gray	<code>Normal exposures</code>	The field is blank
gray	<code>No NightFront config to check against yet</code>	No config could be read, so nothing to check
green	<code>Matches "High Wind" in your config</code>	The name is recognized (matching ignores case)
amber	<code>Not in your config - normal exposures will be used</code>	A typo or a renamed variant
amber	<code>Your config defines no variants - normal exposures will be used</code>	The config has no exposure variants at all

Heads up: An unrecognized variant name doesn't fail the replan. It plans at your normal exposures and says nothing at run time, so a typo means your gusty-hours subs are never shortened on a plan that looks completely normal. The amber dot, and the matching validation issue, are the only warning you get. Check them while you build the sequence.

An exposure variant doesn't change how much sky time a target gets. The same time is divided into more, shorter frames (or fewer, longer ones).

Replan and hand-built plans

NightFront can produce a night in several ways: the optimizer, a plan you built by hand in the desktop app's Editor, or a plan generated by one of its planners (Greedy, Proportional or Dispatch), including the Greedy and Proportional modes on the Options tab. Replan re-solves with the optimizer, so running it over a hand-built or generated night would replace that plan, unattended, in the middle of the night.

So Replan **declines** by default when tonight's plan wasn't produced by the optimizer. It leaves the plan in place, shows a status such as `Left the hand-built plan "Tuesday" in place`, and posts an information notification (not a warning). Tick **Replan hand-built plans** to re-solve such a night anyway.

- **Plans from older desktop builds replan as they always did.** Replan decides from a small marker file that NightFront writes beside each exported plan, and a plan without one is treated as an optimizer plan.
- **A marker that can't be read counts as "not the optimizer"**, so a damaged marker errs toward keeping your plan.
- **A plan generated in Greedy or Proportional mode on the Options tab is declined too.** Those are auto-pilot modes, and Replan stays out of their way.

When Replan doesn't replace the plan

Status message	Container
All targets already complete - nothing to replan	Emptied
Left ... in place	Unchanged (a hand-built or generated plan)
No NightFront config found	Unchanged
Another NightFront CLI run is already in progress	Unchanged
The planner's own message, such as a lost forecast	Unchanged

Note: A replan that can't get a forecast leaves tonight's plan alone rather than emptying the container. The storm that tripped your safety monitor may also have taken the internet down; when it passes, NINA carries on with the plan it had.

Replan runs its solve on a fixed, fast setting (the budget is not adjustable), so a replan typically takes seconds. The plugin waits up to 5 minutes before giving up on it.

Seeing Trigger

Watches the **star FWHM** reported by your connected ASCOM weather device and runs the instructions you drop into it (typically a Replan) when the reading crosses your threshold.

On screen	Default	Meaning
Fire when FWHM <code><=</code> / <code>>=</code>	<code><=</code>	Fire when seeing gets better than (<code><=</code>) or worse than (<code>>=</code>) the threshold
Threshold	2.5"	In arcseconds FWHM
Poll every	5 min	How often the weather device is read
Status line	Not yet sampled	The last reading, for example FWHM 1.84" from the weather device, as of 22:15
Instructions will go here...	empty	Drop the instructions to run

How it decides:

- **It fires on a crossing, not on a level.** The first reading only sets a baseline, so a night that starts already past the threshold doesn't fire the moment the sequence begins.
- **It re-arms after two hours.** If the condition is still true two hours after it fired, it fires again. A long spell of good (or bad) seeing keeps producing fresh replans rather than one at the start.
- **A missing reading never fires.** If the device drops out, the trigger goes quiet rather than treating the gap as a value.

Heads up: This trigger needs a weather device that publishes star FWHM. A reading of 0 is treated as no reading, because several ASCOM drivers report 0 for a sensor they don't have. If your device is connected and reports no FWHM, the sequencer shows an issue saying the trigger can never fire. With no device connected (the usual state while editing during the day) it says nothing.

Live seeing also feeds the planner. When Nightly Update or Replan runs, a current FWHM reading from the device (or, if the device misses that exact moment, this trigger's most recent reading from the last two hours) replaces the forecast seeing for the hours around now. With no FWHM source at all, the plan uses the forecast's seeing. Live cloud cover is read from the weather device either way.

Wind Speed Trigger

Runs **one** set of instructions when wind speed rises to your threshold and **another** when it drops back below. Acting on wind is usually a pair of actions, so it has two drop areas.

On screen	Default	Meaning
Wind speed threshold	20 km/h	The crossing point
Must hold for	0 min	How long the reading must stay on the new side before the crossing counts. 0 is off
Status line	No reading yet	The last reading, for example 24 km/h – at or above 20 km/h
When it rises to the threshold:	empty	Runs when wind reaches or passes the threshold
When it drops back below:	empty	Runs when wind falls back under it

The natural wiring is:

- **Above:** a **Replan** whose **Exposure variant** names a shorter-sub variant, such as "High Wind".
- **Below:** a **Replan** with **Exposure variant** left blank.

Gusty hours then shoot shorter subs, so a gust ruins less, and calm hours go back to normal. Total integration is unchanged either way.

How it decides:

- **The first reading only sets a baseline.** A sequence that starts in wind has crossed nothing, and firing there would replan moments after Nightly Update planned against those same conditions.
- **Only the windy side re-arms.** If it is still at or above the threshold two hours after the last above-threshold run, that side runs again. The calm side runs only on a real crossing back down, so a calm night doesn't replan every two hours.
- **A missing reading runs nothing.** A disconnected device is never treated as calm.
- **"Must hold for" waits the crossing out.** If your wind sits on the threshold and crosses back and forth, each crossing replans — a fresh solve every time. Set this to the number of minutes a change has to last before it counts, and a brief gust and back runs **nothing at all**, in either direction. It is a wait, not a cool-off: it never acts on the gust and then refuses the recovery. Left at 0 it behaves exactly as it always has.

Heads up: This is not a wind cutoff. NINA's safety monitor alone decides whether it's safe to image. This trigger only chooses which of your own instructions run.

Note: NINA checks triggers between sequence items, so with long subframes a crossing can go unnoticed for several minutes, and no trigger can interrupt an exposure already running.

Before Target and After Target triggers

Two triggers that run whatever instructions you drop into them around each NightFront target:

- **Before Target** fires just before a target starts, ahead of its own slew and centering.
- **After Target** fires right after a target finishes, whether it completed, was skipped or failed, and before the sequence moves on. That includes the night's last target.

Their status line reads `Not yet fired`, then `Fired for M 31 at 22:04`.

A typical use is a Ground Station message. Ground Station's `$$TARGET_NAME$$`, `$$TARGET_RA$$` and `$$TARGET_DEC$$` tokens resolve to the NightFront target that fired the trigger.

- **Each fires at most once per target.** A container re-entered by your nightly loop doesn't announce the same target again, and a target NINA is about to skip because its own time or altitude limits have already closed never fires Before Target at all.
- **After Target fires only for a target that genuinely began.**
- **A target a Replan interrupted still gets its After Target.** A mid-night Replan replaces the whole plan, so the target that was running ends at its next step and leaves the sequence. It is reported once, just before the new plan's first target begins, rather than being the one target you hear nothing about.
- **A Replan in After Target skips the target a new plan just replaced.** The night was re-solved seconds earlier from the same progress, so solving it again would only reshuffle the plan. That Replan leaves the new plan in place and says so in its status and the log. Everything else in the drop area still runs, and After Target for any target that ends on its own replans as usual.
- **They fire only for NightFront's own targets.** Other target containers elsewhere in your sequence don't set them off.

Heads up: Place these on the NightFront Container or an ancestor of it. On a sibling branch such as "Once Safe" the sequencer shows: *"This trigger's placement can never see the NightFront Container's targets - it must sit on the NightFront Container itself, or one of its ancestors (e.g. the same branch as 'Loop while safe'), not a sibling branch such as 'Once Safe'."*

While Work Remains loop condition

A loop condition for the outermost container of an unattended sequence: the one that opens the roof each evening. It keeps looping while any **enabled, Active, not-yet-complete** target in your NightFront config still needs integration time, and stops once none do. An observatory left running then stops opening for nights it has nothing to shoot.

It has no fields. Its one line says what the last check found:

Status message	Loop
2 active targets still to shoot (31.4 h remaining) - continuing.	Continues
All 3 active targets are complete - stopping.	Stops
No enabled, incomplete Active targets in the NightFront config - stopping.	Stops
No targets in the NightFront config - stopping.	Stops
No NightFront config found (...) - continuing.	Continues
The NightFront config could not be read - continuing.	Continues
1 target driven by an exposure template, whose remaining time cannot be read here - continuing.	Continues

The remaining time is the same figure as the **Remaining** column on the Options tab's Targets page: what you requested, minus what you accepted, minus what the plugin has collected since.

- **Anything it can't determine keeps the loop running.** A missing or unreadable config, a config path pointing at the wrong `.json`, or a target whose exposures come from a template all continue. Continuing wrongly costs one pass that Nightly Update reports as nothing to plan. Stopping wrongly would end the night at dusk with every instruction below it skipped.
- **It answers as of the start of the observing night.** Frames shot tonight reach the planner by a different route and aren't subtracted here as well, so checking mid-night won't break the loop out from under a plan that's still running.
- **It never interrupts an exposure.** It's checked only between items, so the frame that finishes a project isn't abandoned.

Note: When the condition can tell, while you edit, that it will never be able to stop (no config, a file that isn't a NightFront config, or a template-driven target), it shows an amber warning line under its status. That's advice, not a failure: it never fails the sequence, and it never flags "everything is complete" as a problem. Give template-driven targets their own filter rows in the desktop app if you want the loop to be able to stop.

NightFront Filter Offsets instruction

Measures how far each filter's focus sits from a reference filter, by autofocusing reference, filter, reference over the filters your targets use, and commits the offsets that pass its checks to NINA's filter offset table without waiting for you. Its one field is a **Budget** in minutes; **0** means the evening twilight, Sun -12° to -18° . It needs **Manage filter offsets** and a reference filter on the Options tab, and it exits quietly, with a status line saying why, when either is missing. How it works, what it will and will not write, and the NINA settings to change once are on [Filter offsets](#).

NightFront Save Night's Log instruction

NINA keeps its log file open for as long as it runs, so a copy or sync of the live file can lag hours behind, and the NightFront app's log analysis then sees only part of the night. This instruction writes a closed copy of **this observing night's** part of the log, sky flats included, while NINA stays open.

Put it last in your End area, after sky flats, park and warm-up, so everything the night did is in the copy. In a sequence that loops once per night, that means last in the end-of-night steps inside the loop, as in the [example night](#). It has no fields of its own.

When it runs it:

- takes the night that began at the most recent local noon (run at 07:30 on October 4 or at 23:00 on October 3, it saves the night of October 3). A run **between noon and dusk** (sunset at your NINA profile's location, or 18:00 local without one) saves the night that just ended instead, from yesterday's noon through now and named for yesterday; its status message says "for the night of ";
- reads every log in NINA's log folder written since that noon, keeping each log's header and every entry from noon onward, with any error details that follow an entry;

- writes one file per log into a **nina-logs** folder beside the NightFront config the plugin plans from (or in the data folder, when no config is set), named after the original log with `(night yyyy-mm-dd)` added. Running it again for the same night replaces that night's copy.

NINA's own log folder is only read: nothing there is changed, moved or deleted. There is nothing to set up in the NightFront app: the logs sit beside its config, so the app's **Analysis** page reads the saved nights from this **nina-logs** folder on its own. On that page one **Analyze** button reads the **Most recent N nights** (7 by default) or the nights you choose with **Pick nights...**, and the **Last night · All nights** toggle on the Last night chip switches between the newest night and all of them. See the NightFront User Guide's Analysis page for the details.

Before it cuts, the instruction waits for NINA's log to catch up. It writes one marker line through NINA's own logger, then pauses, **up to 10 seconds at shutdown**, until that line is on disk, so the last steps before it (the park, the camera warm-up) are in the copy. The marker line itself is never copied. NINA writes it at the Info level, so keep NINA's log level at **Info or more detailed**: at a quieter level the marker is never written, and every run waits the full 10 seconds and reports the message below.

The instruction never fails your sequence, since it runs while the system is shutting down. If no folder is set, NINA's log folder can't be found, or a file can't be written, its status line says what went wrong and NINA's log records a warning; the sequence carries on.

The status line can also say *NINA's log did not show this step's end-of-night marker within 10 s, so the last lines before it may be missing*. That is not a failure: the copy is still saved, and the status dot stays **green**, so read the message rather than the dot. It means the last few lines before the end of the night may not be in the copy.

Example sequence and templates

The plugin installer puts a **NightFront NINA Templates** folder on your Desktop. It holds one example sequence and five templates that show where each NightFront instruction, loop condition and trigger goes. They use only NINA's own items and NightFront's, and notes (NINA's **Annotation** instruction) say what each part does, why it sits where it does and what not to move.

File	What it is
NightFront Example Night.json	A whole sequence: one night, repeated each night while NightFront has work to do
NightFront - Start of Night.template.json	Your startup, then NightFront Filter Offsets and Nightly Update
NightFront - Imaging.template.json	The NightFront Container in a loop-while-safe container, with its triggers
NightFront - Safety Recovery.template.json	Parks, waits out unsafe weather, unparks, then replans
NightFront - Sky Flats.template.json	Dawn sky flats matched to the night's lights
NightFront - End of Night.template.json	Your shutdown, then NightFront Save Night's Log
README.txt	How to use them

Heads up: Copy a file somewhere else before you edit it. The installer replaces this folder on every upgrade, and uninstalling the plugin removes it, so anything you change there is lost.

Containers named "Your own ..." are placeholders for your rig's own steps: cooling, opening the roof, your first autofocus, closing up. The examples do only the Park Scope and Unpark Scope pairs for you: each Park Scope they ship is followed by an Unpark Scope before the next imaging. Every NightFront item keeps its default settings; the notes say what each field means.

Using them

- **To read the example**, load `NightFront Example Night.json` in NINA's Advanced Sequencer with the button that opens a saved sequence. It needs no equipment connected.
- **To use the templates**, copy the five `.template.json` files into NINA's template folder, the one set under **Options** → **General** → **Sequencer** (by default `Documents\N.I.N.A\Templates`). They then appear in the sequencer's **Templates** list, ready to drag into a sequence of your own (restart NINA if they don't).

The files need the NightFront plugin installed. Without it, NightFront's items load as "Unknown Instruction".

The example night

```
NightFront Example Night
├─ Start: notes only
├─ Targets
│   └─ "Every night: loops while NightFront has work to do" ← condition: While Work Remains
│       ├── NightFront - Start of Night
│       ├── "Imaging until dawn: ..." ← condition: Loop until nautical dawn
│       │   ├── NightFront - Imaging
│       │   └─ NightFront - Safety Recovery
│       ├── NightFront - Sky Flats
│       ├── NightFront - End of Night
│       └─ "Wait for sunrise before the next night"
│           └─ Wait for Time: sunrise, offset +5 minutes
└─ End: notes only
```

While Work Remains repeats the night once per night while any enabled Active target still needs time, and stops when none does. The End area then runs; in the example it holds only a note.

The five templates

NightFront - Start of Night

```
NightFront - Start of Night
├─ Wait for Time: sunset
├─ "Your own startup goes here: connect, cool the camera, open the roof"
├─ Wait for Time: nautical dusk
├─ "Your own first focus goes here: unpark, slew to a star field, autofocus"
│   └─ Unpark Scope
├─ NightFront Filter Offsets (optional; Budget 0 = the evening twilight)
└─ Nightly Update (If generating, plan for up to 180 s)
```

- **Unpark Scope is already in the first-focus step.** End of Night parks the mount, and a parked mount refuses every slew, so without it the second night would image with the mount parked. If your roof must close over a parked mount (a roll-off roof), open it BEFORE Unpark Scope.
- **Nightly Update runs once a night, before the NightFront Container**, and after anything that spends the evening, so a plan it generates starts when imaging really can.
- **NightFront Filter Offsets** does nothing until you turn on **Manage filter offsets** and choose a reference filter on the Options tab. It never slews, so your first focus step points the telescope first. See [Filter offsets](#).

NightFront - Imaging

```
NightFront - Imaging ← condition: Loop while safe
├─ triggers: Meridian Flip, Autofocus after HFR increase, Autofocus after temperature change,
│           Autofocus after filter change, Center After Drift, Restore Guiding (NINA's);
│           Before Target, After Target, Seeing Trigger, Wind Speed Trigger (NightFront's)
└─ NightFront Container (empty: Nightly Update fills it)
```

- **The NightFront Container is empty on purpose.** Nightly Update fills it with tonight's targets and Replan refills it. Never put anything in it yourself.
- **Before Target and After Target sit on this container,** above the NightFront Container, as they must. Each has a note where your notification goes.
- **Seeing Trigger and Wind Speed Trigger each hold a Replan.** The Wind Speed Trigger has one in each drop area: type a shorter-sub exposure variant's name into the upper one's **Exposure variant**, or leave both blank.
- **Center After Drift sits here, above the NightFront Container,** where the plugin points it at each target as it starts.
- **Autofocus after filter change stays** until every filter has a filter offset. Remove it only when you turn on NINA's **Use filter offsets** ([Set up NINA once, at the end](#)).
- **It needs a safety monitor connected in NINA.** Without one, Loop while safe never lets it run.

NightFront - Safety Recovery

```
NightFront - Safety Recovery
├─ "Your own weather-stop steps go here: park, close the roof"
│   └─ Park Scope
├─ "Wait while unsafe"    ← condition: Loop while unsafe
│   └─ Wait for Time Span: 60 s
├─ "Your own recovery steps go here: unpark, reopen the roof, cool the camera"
│   └─ Unpark Scope
└─ Replan
```

Put it right after **NightFront - Imaging**, both inside a container that loops until dawn, as the example does. When the safety monitor ends Imaging, this parks the mount, waits for safe weather, unparks the mount, runs your recovery steps and replans, and the loop goes back to Imaging with a plan that fits the time that is left.

- **Park Scope comes before the wait, never inside it.** Inside, Loop while unsafe would run it every minute. Add closing your roof or dome beside it; your recovery steps undo both.
- **Unpark Scope comes after the wait, never inside it,** and before the Replan. A parked mount refuses every slew, so without it the loop would go back to Imaging and every target would fail until dawn. If your roof must close over a parked mount (a roll-off roof), reopen the roof BEFORE Unpark Scope, just as you close it AFTER Park Scope.

NightFront - Sky Flats

```
NightFront - Sky Flats    ← conditions: NightFront While Flats Remain, Loop while safe,
                          Loop until sunrise, offset +30 minutes
├─ Wait for Time: sunrise, offset -30 minutes
├─ Unpark Scope
├─ "Rotate and slew together"  (parallel)
│   └─ NightFront Rotate to Next Angle
│       └─ Slew to Alt/Az: altitude 80°, azimuth 270° (west, away from the rising Sun)
├─ Set Tracking: stopped
├─ "Every flat still needed at this angle" ← condition: NightFront While Same Rotation
│   └─ NightFront Sky Flats
```

- **With nothing outstanding it is skipped.** NightFront While Flats Remain is false when no flat is outstanding.

- **The Loop until condition is the time limit, and it must stay.** Once the sky is too bright, NINA's Sky Flat fails and the flat is handed back as still needed. Without a limit the loop would try it again, and alert you again, all day, and the next night would never start. A set the limit cuts off is handed back and shot on a later dawn.
- **Move the sunrise offsets** to suit your filters and NightFront Sky Flats' **Max exposure:** the wait's sets when flats start, the limit's when they stop. **Flat Filter Order** on the Options tab sets the order the filters are shot in.
- **With a flat panel,** use NightFront Trained Flats in place of NightFront Sky Flats, and drop the wait, the slew and the tracking change. Keep a time limit, moved to suit.
- **If your safety monitor reports unsafe in daylight,** remove Loop while safe from this container, but keep the time limit.

See [Calibration flats](#) for how the flat requirements build up.

NightFront - End of Night

```
NightFront - End of Night
├─ Park Scope
├─ "Your own shutdown goes here: close the roof, warm the camera"
└─ NightFront Save Night's Log (last)
```

Why it is laid out this way

- **Stock NINA has no "Once Safe" instruction.** Safety Recovery builds the wait from NINA's own **Loop while unsafe** condition around a one-minute wait. While the sky is safe that container is skipped entirely.
- **Replan comes after Imaging, never before it.** Imaging loops while the sky is safe, so on a clear night it runs until dawn and Safety Recovery is never reached. A Replan before Imaging would re-solve, on every first pass, the plan Nightly Update had just made. After it, a Replan runs once per weather stop. The cost: if Imaging ever ends before dawn for another reason (you skip it, say), that also parks and replans once. At dawn the loop's time condition skips Safety Recovery.
- **One wait at each end of the night.** Start of Night begins with a wait for sunset, and the example ends each night with a wait until 5 minutes after sunrise. Together they make each pass of the nightly loop start in the evening. Without the sunrise wait, a night that ends before sunrise (no flats still needed, say) would run the next night's startup at dawn. NINA reads a wait for sunset run after midnight and before sunrise as already past, so a sequence started in the middle of the night goes straight on.
- **The sunrise wait needs its offset.** NINA ends a wait on a whole second, but compares the next wait for sunset against sunrise to a fraction of a second. A wait for exactly sunrise usually ends a moment early, and the next night's startup then runs at dawn anyway.
- **Far north in summer, swap the nautical items.** Above roughly 54° latitude in summer the Sun never gets 12° below the horizon, so NINA has no nautical dusk or dawn and those waits and loops fail validation. Use a fixed time, or sunset and sunrise, instead. Astronomical dusk and dawn are no help: they vanish even sooner, above roughly 48°.
- **End of Night sits inside the nightly loop,** so flats, park and **NightFront Save Night's Log** happen every night rather than only when the whole sequence ends. Save Night's Log is the last step that does anything;

only the wait for sunrise follows it.

- **The plugin's defaults are kept:** a 180 s Time Budget, a Filter Offsets budget of 0, a Seeing Trigger at 2.5" or better, a Wind Speed Trigger at 20 km/h with no hold time, and NINA's own trigger defaults. Change them in place once you know your site.

Note: The files load in NINA 3.0 and later, the plugin's minimum. Before they shipped they were checked against NINA 3.0's own instruction list and loaded through NINA's own sequence reader; the plugin's tests keep checking them on every build.

The Options tab

NightFront's settings live in NINA under **Options** → **Plugins** → **NightFront**, on three pages:

- **Setup**, which you configure once.
- **Targets**, where you sit at the imaging computer and see, and adjust, what tonight is asked to shoot.
- **Filter offsets**, the measured focus offset of each filter, any parfocal groups you choose, and the history of what NightFront wrote into NINA's table. It has its own page, [Filter offsets](#).

These settings are global to the NINA installation, not per equipment profile. The sequence items themselves are described in [Sequence instructions and triggers](#).

Options tab Setup page

The page opens with a short introduction to NightFront and a **Plugin Guide** button, which opens this guide (a PDF installed with the plugin). If the file is missing, the button says where it should be.

Setting	Default	What it does
NightFront Data Folder	clear (the config's folder)	Where plans, the collected-frames ledger and calibration metadata live. Leave it clear to use the NightFront config's own folder; Clear goes back to that.
NightFront Config (for online planning)	not set	The NightFront config the plugin plans from. Browse... to pick one, Clear to go back to the fallback. Set this or the data folder; normally just this.
Planning Mode	Optimizer (GA)	How Nightly Update builds a night when it generates one
Plan Selection	Balanced	Which schedule to take from the optimizer's range of them
Fill the rest of the night when the plan finishes early	on	Replan by itself when tonight's plan runs out with darkness left
Custom Horizon (from NINA)	read-only	The horizon file NINA's profile names, if any
Flat Filter Order (labeled with an example, L, B, G, R, OIII, Ha, SII)	blank	The order calibration flats are shot in
Flat Refresh Days	30	How long a completed flat stays done before it can be reshot
Plan Retention Days	2	How many days of past plan files to keep in the data folder

Plugin data folder

Leave **NightFront Data Folder** clear and the plugin uses the folder that holds your NightFront config; the row then shows that folder. Export nightly plans from the desktop app into the same folder. Set the data folder only to keep the plugin's files somewhere else, and **Clear** it to go back to the config's folder. **NightFront Save Night's Log**

saves the night logs into a `nina-logs` folder beside the config either way, which is where the desktop app's **Analysis** page reads them. The plugin keeps its own files in the data folder:

- `NightFrontPlan.json`, the plan Nightly Update generated,
- `calibration.metadata.json`, which holds the calibration-flat requirements and the collected-frames ledger,
- `replan-history`, a folder of plans that a Replan replaced.

Each Nightly Update tidies old plan files, their progress files and replan-history copies older than **Plan Retention Days**. Set it to 0 to keep everything.

The plugin helps keep the two settings together:

- With a data folder set, **Browse...** for a config in another folder and the plugin asks whether to make that folder the data folder too.
- **Browse...** for a data folder holding one NightFront config and it offers to plan from that config. When a config is already set, closing that question without answering counts as **No**, since **Yes** changes which config tonight plans from. A folder holding several configs (a sync client's conflict copy of one counts) gets a note naming them all; choose the right one with the config's **Browse...**
- Any change that would leave `calibration.metadata.json` behind asks first: a new data folder, a **Clear**, or a new config while the data folder is clear and following it. Closing the question without answering keeps the file in use. That file doesn't move, so from a folder without it the plugin starts with an empty collected-frames ledger and no queued flats: frames you already shot are planned again. Copy the file across first if you mean to move.
- An amber line under the two settings, and the same note on Nightly Update in the sequence, says when the config is in another folder than a data folder you set, or when the config setting names a file that no longer exists. It is a warning only: the night still plans.

Which NightFront config the plugin reads

Point **NightFront Config (for online planning)** at the `.json` the desktop app's Save writes. The plugin then plans from your current accepted counts.

If it's left blank, or the file it names no longer exists, the plugin falls back to the `session-config.json` that an export from the desktop app leaves in the data folder. Nightly Update, Replan, While Work Remains, the Replan exposure-variant list and the whole Targets page all use the same resolution, so they always describe the same config.

Heads up: The exported `session-config.json` is frozen at export time. Its accepted counts never move, so a stale copy re-schedules data you already have while reporting success. Setting the config path avoids that. The file dialog can't return "nothing", so **Clear** is the only way back to the fallback once you've picked a file.

How to tell how old the config is. The Targets page's status line reports `last saved N days ago` after a **Reload**, and Nightly Update shows an amber advisory in the sequencer once the config is 7 or more days old. Neither ever stops a night.

Planning Mode setting

Chooses how **Nightly Update** plans a night when there is no plan file and it has to generate one.

Option	What you get
Optimizer (GA)	The default. Searches for the best trade between time on sensor and image quality, within Nightly Update's Time Budget, and takes the point Plan Selection names.
Greedy	Fills the night with the highest-priority target that is feasible and still needs time, then the next, following the target priority order you set in the desktop app. Instant and deterministic, and only as good as that order.
Proportional	Splits each night evenly between the targets that can use it, so everything on your list advances together instead of the top of it taking every clear night. Priority order decides who goes first among targets behind their share.

What changes under **Greedy** or **Proportional**:

- **Plan Selection has no effect**, since each produces a single plan rather than a range. The page adds the caption — *not used under Greedy or Proportional planning*.
- **Nightly Update's Time Budget has no effect**, since there is no search. The page says so in a second caption.
- **Frames already collected are still subtracted**. Nothing about the ledger changes.
- **Replan leaves those nights alone**. Replan re-solves with the optimizer, and it declines to overwrite a generated plan unless you tick **Replan hand-built plans** on the instruction.
- **Replan itself still uses the optimizer**, so a night exported from the desktop optimizer stays replannable whatever this setting says.

Neither setting is cleared or grayed out when you switch; both apply again the moment you switch back to the optimizer.

Plan Selection setting

The optimizer produces a whole range of schedules, trading time on sensor against image quality. **Plan Selection** chooses which one tonight gets:

Option	Takes
Max Quality	The best-quality end: fewer hours, better placed
Balanced	The knee of the utilization/quality curve (the default)
Max Utilization	The most sky time used, accepting some of it at worse airmass

It applies to both Nightly Update and Replan, so a mid-night replan can't quietly switch philosophies on you.

Fill the rest of the night setting

On by default. When the NightFront Container's plan finishes while usable darkness is left, the container replans the rest of the night by itself and keeps imaging; if every enabled Active target is already complete, it keeps imaging the best-placed one with extra subs. Turn it off and the container simply ends with its plan, as before. The rules (how much darkness counts, the limit of 3 refills a night, hand-built plans) are in [When the plan finishes early](#).

Custom horizon readout

Custom Horizon (from NINA) shows the horizon file set in NINA's own **Options → Astrometry → Custom Horizon**, or, when there isn't one: *No custom horizon configured in NINA — NightFront will use the horizon in its own config, if that config states one.* You can't edit it here; change it in NINA.

A horizon stored in your NightFront config takes precedence over this one. NINA's horizon only fills in for a config that doesn't carry its own. See [Custom horizon from NINA's profile](#).

Flat settings on the Setup page

Flat Filter Order and **Flat Refresh Days** drive the calibration-flats instructions. Both are explained on [Calibration flats in NINA](#).

Options tab Targets page

Shows where every target stands, without a trip back to the desktop app: an **Active** column and a **Planned** column side by side, letting you change how many subframes tonight is asked for, move a target between the two, enable or disable one, and send a target to NINA's Framing Assistant to see it framed.

Nothing is read until you press **Reload**. The line under the buttons names the config being shown, and the status line summarizes it, for example `4 targets, 17 filter rows • last saved 3 days ago.`

Target and filter rows

Each target is **one collapsed line**: its name, how complete it is, and the hours it still needs, for example `M 16 62% complete • 4.5 h remaining`. Click it to expand its exposure plan, one row per filter:

Column	Meaning
Filter	The filter
Requested	How many subframes you want in total. The only editable field.
Accepted	What you kept when you last reviewed, in the desktop app
Collected	What this plugin has counted since that review
Remaining	What is still needed, in subframes
Rem (h)	The same, in hours
Exp	The sub length, in seconds
Moon Profile	The moon avoidance profile that filter uses

- **A mosaic is one row**, named for the mosaic and captioned with its panel count, with its panels nested underneath. Its completion and hours are pooled across panels in seconds, so a panel with twenty hours left isn't averaged against one that's nearly done.
- **The completion percentage is derived from the Remaining column**, so the headline and the numbers under it can't disagree, including on a filter with a partial-exposure residue that still costs a whole sub.

- **A filter can appear twice for one target** when you've planned it at two sub lengths. They're two separate amounts of sky time, so they stay two rows.
- **A target you've marked complete** in the desktop app appears in neither column.
- **A disabled Active target still appears**, captioned `disabled`, so you can see why it gets no time.
- **A target whose exposures come from an exposure template** shows its heading only, captioned `from template "..."` — not editable here.

The Planned column, and moving a target

Every target you haven't marked complete is in one column or the other. A mosaic is placed by whichever column holds the **majority of its panels** — you split one only by moving individual panels in the desktop app — and its heading names the split, for example `2 of 4 panels Planned`.

Planned rows are read-only here. They still show Requested / Accepted / Remaining, but you can't type into them; move the target to Active first, then edit it there.

Each row's heading carries a button:

- **→ Planned / → Active** moves a single target, or a whole mosaic (every panel moves together — there's no per-panel Move here, on purpose: a half-moved mosaic would schedule some panels and not others while looking like a normal, healthy plan).
- **Enable / Disable** on a single target, **Enable all / Disable all** on a mosaic root. Per-panel disable stays a desktop-only control for now.

Both write straight to your config, the same way Save does — there's nothing to save afterward, and nothing is undone by pressing Reload. Both wait their turn behind a planning run in progress exactly as Save does, and refuse the same way if it's still busy after a minute.

Neither takes effect in the sequence already running — a target you move or disable at dusk is still part of tonight's plan once NINA has imported it. The *next* Nightly Update or Replan is what picks up the change. **While Work Remains is the one exception:** it re-reads the config live, so disabling the last Active target with work left can end an unattended run at its next check.

Moving or enabling/disabling never touches priority order, demand, accepted counts, or a target's coordinates — only where it's listed and whether it runs.

Every change goes only to the config the page is showing. If you pick a different config (or data folder) on the Setup tab, or press Clear there, after the page was loaded, Move, Enable/Disable and Save refuse and ask you to press **Reload** first — the rows on screen came from the other file, and a change sent to the new one could land on a different target that happens to share a name. Save keeps your typed edits when it refuses, so you can point the Setup tab back and save them. None of them, nor Create or Update below, will write to the `session-config.json` export snapshot the page falls back to when no config is set: the desktop app never opens that file, and its next export would silently undo the change.

Editing Requested and saving

Type a new **Requested** count and press **Save**.

- **The edit goes through NightFront's own planner program**, exactly as a desktop Save would, and nothing else in the config file is touched.
- **An edit inside a collapsed target** adds `edited` to its heading (and to its mosaic's) so it can't sit forgotten.
- **Save waits its turn.** If Nightly Update or Replan is mid-solve, Save waits for it rather than racing it. If it's still busy after a minute, you'll see *NightFront is busy planning right now. Nothing was saved — try again in a moment.*
- **The page always re-reads the file after a Save**, successful or not, so it never shows a number that wasn't written. **Reload** discards unsaved edits, and also discards a framing preview — the targets it offered to update were ranked against the config that was loaded before it.

Accepted stays read-only. It records your judgment about frames you've looked at, and it's one of the two signals that tell the plugin its own tally has been superseded (see [The plugin's collected-frames ledger](#)). A second way to change it would quietly fight the first.

Check tonight button

Press **Check tonight** to ask the planner, target by target, what tonight's sky allows. The answer is added to each collapsed heading, stamped with the time you checked:

Caption	Meaning
<code>imageable 4.2 h</code>	Feasible for that long tonight
<code>imageable 4.2 h, but never 30 min unbroken (best 22 min)</code>	Feasible time exists but never arrives in one stretch long enough to start on. The one way a target looks healthy and still gets nothing all night.
<code>imageable 3.1 h · blocked: Ha, OIII</code>	Feasible, but those filters are ruled out tonight (typically by the moon)
<code>never high enough tonight</code>	Never clears your altitude limit or horizon
<code>too close to the moon tonight</code>	Blocked by moon avoidance
<code>high, or clear of the moon, but never both at once</code>	High enough at some point and clear of the moon at some point, but never at the same time
<code>nothing left to shoot</code>	No remaining demand

A mosaic reports a count instead, such as `3 of 4 panels imageable tonight`, since panels sit a few degrees apart and routinely differ. If some imageable panels never get a long enough stretch, it adds that too: `, 1 not for long enough unbroken`.

When the check can't diagnose targets, the status line says why: *No forecast available, so tonight's sky could not be checked*, *No astronomical darkness tonight.*, or *Every Active target is already complete*.

- **It only reads.** Nothing is written, and your collected-frames tally is not touched.
- **It uses the forecast cache.** A cached forecast is reused while it's fresh: up to 3 hours for 7Timer (the free default) and up to 6 hours for Astrospheric. Otherwise it fetches a new one; an Astrospheric fetch costs API credits, a 7Timer fetch is free. The cache is kept per source, so switching sources doesn't throw away the other's.
- **Your answer survives a Save or Reload.** The time stamp shows how old it is.

- It uses **NINA's custom horizon** the same way planning does.

New target from the Framing Assistant

Frame an object in NINA's Framing Assistant (a single pointing or a mosaic grid), then on the Targets page:

1. Press **Preview**. It reads what's framed and your config's exposure templates, and describes exactly what would be created, for example: *"M 16" — 4 panels (2×2, 20% overlap), named "M 16 Panel 1" ... "M 16 Panel 4", from template "Narrowband". Panel 1 at RA 274.700°, Dec -13.807°, rotation 0.0°.*
2. Pick an **Exposure template**, and check the **Name** box beside it. Preview fills it in from the Framing Assistant's target name; change it here if that's not what you want the new target called. Preview's line always quotes the name in the box, so it says exactly what will be written.
3. Press **Create**.

⚠ **After a Frame, the Framing Assistant's name is the framed target's**, so the next Preview prefills that name. If you then change the framing to make something new, type the new name before Create — the line says *"came from the last Frame"* when that's where the name came from, and Create stays unavailable while the name is already taken.

Each panel becomes a real target in your config, and it:

- carries the template's filter rows written into it directly, with their gains, dither cadences and moon avoidance profiles, so it reconciles against your reviews like any other target,
- lands **Active** and enabled, ready for tonight,
- gets a minimum imaging time of 60 minutes for a single pointing, or 30 minutes per mosaic panel,
- takes the name in the **Name** box, with each mosaic panel numbered after it (`M 16 Panel 1`).

Coordinates are converted to J2000 on the way in. NINA's framing surfaces can hold coordinates for the current epoch, and the difference is about the size of one mosaic panel step: enough to frame a mosaic that looks right and is systematically off.

Preview is where anything gets refused, so Create is never armed for something it would reject:

- *Nothing is framed in the Framing Assistant yet.*
- A blank **Name** box, or a name already used by a target or mosaic in this config (or by a panel the new mosaic would create). Compared exactly, capitals included. Type another name and Create becomes available — no need to preview again.
- A grid set to, say, 2×2 that has only produced one rectangle so far. Let the framing finish computing its panels, then preview again.
- *That config declares no exposure templates.* Add one in the desktop app first.
- No config set on the Setup tab, so the page is showing the `session-config.json` export snapshot. Set your config there first.

Create itself can still refuse:

- **The Setup tab's config changed since the preview.** The preview is discarded; press Preview again.

- **Unsaved Requested edits block Create.** Save or reload them first; creating re-reads the config and would discard them.
- **A name that already exists refuses the whole set.** If any name it would create is already in the config, nothing is created at all, not even the other panels. Two targets sharing a name don't error anywhere downstream; they merge in the NINA export and pool their frame counts. Rename in the desktop app, or use a different mosaic name.

Preview and Create only read the Framing Assistant; neither Preview nor Create ever changes what's framed there. (**Frame**, described below, does — that's its whole job, and it's a separate button you have to press on purpose.)

Frame: see a target in NINA's Framing Assistant

Press **Frame** on any single target, any mosaic panel, or a mosaic's own heading, and NINA's Framing tab opens showing it — centered, at the right rotation. Nothing in your config changes; this only tells NINA where to point its preview.

- **A single target or one panel** frames as a single pointing.
- **A mosaic's own Frame button reproduces the WHOLE grid** — every panel, at the stored center, rows × columns, rotation and overlap — not just one panel. If the stored panels don't add up to a grid NINA can draw exactly (see below), it refuses rather than draw something close, and says why.
- Frame is disabled, with a tooltip, on a target with no coordinates stored.
- Framing may switch NINA's own Framing tab off a cached or offline sky-image source onto an online one, the same way choosing an object there yourself does. Nothing about the target is affected; it's purely NINA's own default for its next preview image.
- **⚠ NINA's own Load Image button (or an automatic refresh) later resets the rotation it's showing back to whatever you last had it at.** Frame's push is a one-time snapshot, not something NINA remembers going forward. If you want the target's framing back, press Frame again rather than Load Image. Update refuses a framing NINA has reset this way (see below); turning the frame yourself afterwards is fine.

A whole-mosaic Frame can refuse, always with a reason rather than an approximate picture:

- *The grid doesn't fit your telescope and camera as NINA currently has them configured* — the panel spacing NightFront stored implies an overlap NINA's field of view can't reproduce with your rig's numbers. A mosaic the current desktop app lays out is placed the same way NINA itself places panels, so this is rare for one; it is most likely on a wide mosaic laid out by an older NightFront, and a very wide new one can still meet it, because NINA works out a camera's field of view slightly differently from NightFront.
- *The panels don't agree on rotation* — check *Preserve Alignment* in NINA's Framing Assistant. Each panel of a mosaic laid out with that option on gets its own rotation, which this plugin can't draw as one grid.
- *This mosaic's panels were laid out by an older NightFront or an AstroPM import* — at an angle that isn't a multiple of 90°, such a mosaic is the mirror image of the grid NINA draws at the same rotation, so NINA's panels would land away from yours. Frame refuses rather than draw it wrong. Correct the mosaic in the NightFront desktop app to frame it whole; per-panel Frame still works on each panel individually. Mosaics created in NINA, and ones the current desktop app lays out, are not affected. If only NINA's camera could tell the mosaic was mirrored, because the NightFront config states no sensor size, the message asks you to

enter the sensor size on the desktop app's Imaging System page first: the desktop app cannot offer the correction until it knows the camera.

- *Its panels fit NINA's grid both ways round ... State the camera's Sensor width (px) and Sensor height (px) — at 45°, 135°, 225° or 315° an older NightFront's mirrored layout and NINA's own fit the same panels, with rows and columns swapped, and only the camera's shape tells them apart. For a single row or column it is the camera's field of view instead, which also needs Focal length and Pixel size. Frame sends NINA's camera to NightFront for this, but the NightFront config's own optics win whenever it states them. The message says which is missing or which disagrees: enter the camera on the desktop app's Imaging System page, or connect the camera in NINA so its size reaches NightFront. When the camera is known and fits neither way round, the message says the mosaic was laid out for another camera, or the stated optics are wrong — or, on a very wide field (a camera lens covering more than about 23°), that an older NightFront laid it out from its slightly narrower field; frame its panels one at a time, or create the mosaic again in the desktop app.*
- *Its panels were laid out mirrored, within 0.1° of a right angle — so close to a right angle that the frames sit at most 0.2° askew and there is nothing to correct (the desktop app doesn't flag it), but the grid is wide enough that NINA's own grid at the stored rotation misses a panel past the 5" check. Frame its panels one at a time.*
- *NINA could draw a panel up to N" from where the config has it — the grid NightFront recovered already sits a little off that panel, and NINA's one overlap adds a little more; together they pass the 5" check, so the frame could not be confirmed. Frame refuses before pushing anything. Frame its panels one at a time.*

Frame → Preview → Update is protected the same way a normal re-point is (see below): if the sky image NINA needs to confirm the framing never loads, Update on that target stays disabled until you frame it again successfully — NINA can otherwise report success on a framing that never actually arrived, and Update is the one action here that discards accepted subframes.

Update is also refused, until you press Frame again, when **NINA itself changed the framing after a Frame** without you moving it:

- *NINA reloaded the sky image and put back its last-used rotation — its Load Image button, or an automatic refresh, drew the target at the rotation you last used in NINA rather than the target's. Update would have written that rotation over the target. Turning the frame yourself is not affected: NINA remembers an angle you set, and that is how the two are told apart.*
- *NINA's camera or focal length changed since the Frame — for a whole mosaic, a camera connecting (or its size changing) re-spaces the panels around the same center, so they no longer match the stored ones.*

Dragging the frame to somewhere new makes it your framing, and Update works on it as usual.

Update an existing target from the Framing Assistant

The same framing can also **re-point a target you already have**, instead of creating a new one — for when a mosaic was laid out slightly off, or you want to recentre a field you have been shooting.

Press **Preview** as above. Beside Create you will find **Or update an existing target:** — a list of this config's targets, **nearest first** to what you framed, with a mosaic appearing once and ranked by its closest panel. If the framing carries the name of a target already in this config (as it does after you press Frame), only that target is picked for you, and nothing is picked if it can't be updated from this framing. **After framing one mosaic panel, nothing is picked**, and the line says why: a mosaic is re-pointed only as a whole, so frame it from its own heading instead. Otherwise, for an object you found in NINA yourself, the nearest one whose panel count matches your framing is

picked for you — and where two sit at the same spot, whichever has the least accepted integration, so the default is the one with the least to lose. (Duplicating a target makes exactly such a pair: the copy carries the plan and none of the progress.) Choose another if you want; then press **Update**.

The confirmation line above the buttons says what Update would do before you press it, including the part that cannot be undone:

Update would re-point 'Barnard 150' to this framing. Any panel that actually moves loses its accepted subframe counts and is stamped reviewed.

Frames collected at one pointing do not belong to a different composition, so every panel that genuinely moves has its **Accepted** counts zeroed and is stamped **Reviewed** — the same rule the desktop app applies when you move a target by hand or re-frame it on the Sky map. Planned is left alone, so the remaining time goes back to the full plan. A panel that did not really move — a shift under 5 arcseconds with rotation within 0.1° — is left exactly as it was, counts and all, and the result line says how many were left alone.

The frames themselves stay in your Projects tree, and the desktop app's next **Accept lights** will offer to count them again. To make the reset stick, move that target's old lights into the rig's **rejected** folder.

Your framed rectangles are matched to the target's existing panels **by nearest coordinate**, never by the order they were drawn in — so a mosaic re-framed with its panels in a different order still lands panel-for-panel.

Update refuses anything it is not sure about, and changes nothing at all when it does:

- **The panel counts must match.** A two-panel mosaic against a single framed rectangle is refused; adding or removing panels is not an update. The picker will not pre-select a candidate of the wrong shape, and Update stays disabled while one is chosen.
- **A completed target is refused.** Clear its completion in the desktop app first.
- **A framing aimed somewhere else is refused.** If any panel would move more than 5° , Update refuses rather than treating it as a very large recomposition.
- **A target with no coordinates is refused** — there is nothing to match a rectangle against, and Update will not go and look one up.
- **A name that matches two targets is refused**, rather than one being picked for you.
- **A single panel of a mosaic cannot be updated on its own** — that would leave its siblings where they were, which is not a framing you could have drawn.
- **Unsaved Requested edits block Update**, exactly as they block Create: updating re-reads the config and would discard them.
- **A framing that is already where the target is** reports that nothing needed doing.

If the picker is empty, the line above it says why. An empty list with nothing said means there is genuinely nothing near what you framed. The other likely cause is an older `nightfront-cli.exe` beside a newer plugin — the line names that case, and the remedy is to update the plugin, which brings its own copy.

The plugin's collected-frames ledger

The plugin counts the frames it actually shoots, and the planner subtracts them from what each target still needs. That's the **Collected** column. It means several unattended nights can run back to back without you reviewing subs in between, and the plan still moves on to what's left:

Remaining = Requested – Accepted – Collected

The ledger lives in `calibration.metadata.json` in the data folder. It's counted in seconds of exposure, so a night shot at an exposure variant's shorter subs still subtracts the right amount; the **Collected** column converts that back to frames at the row's own sub length.

What clears the collected-frames ledger

When you review a target in the desktop app, the plugin's tally for that target is superseded and its rows are **cleared outright**. It doesn't subtract the frames you accepted; it drops the whole count, because your review is now the record. Either of these counts as a review, per target:

- **the target's review stamp changes**, for example when you press **Mark reviewed** on it in the desktop app (other review actions there stamp it too), or
- **its Accepted count changes** in the config.

The review stamp is the one that matters most: a review in which you reject every frame leaves Accepted unchanged but still clears the tally, so rejected frames stop being subtracted and get re-shot.

- **Tonight's rows are never cleared**, whatever triggered the clear. Frames from the night in progress stay counted until the night is over.
- **The first time the plugin sees a target, it clears nothing**. It only notes the current Accepted count to compare against later.
- **The Targets page's status line explains an empty Collected column**, for example `no collected-frames ledger yet`, or `no data folder set`, so collected frames are not being counted.

Renaming a target

Each target carries a hidden identity, and the ledger follows that rather than the target's name — so renaming a target in the desktop app keeps its collected frames, and reviewing it afterwards clears them as it should.

Heads up: This protects frames counted from this version onward. Anything already in your `calibration.metadata.json` from before it carries no identity, so a rename still orphans those, and they get re-shot. The same goes for any night a plugin older than this one recorded.

Heads up: It needs the desktop app to be recent enough to stamp its targets. An older NightFront simply means no protection — nothing breaks. An older plugin beside a newer app ignores the identity and behaves exactly as it always did.

If a rename does orphan something, the CLI's log records that a ledger entry matched no target. That line is the only signal, so it is worth knowing it exists.

Heads up: The data folder and config path are the same for every NINA equipment profile. If two rigs run from one NINA installation, they share one ledger and subtract each other's frames. The plugin is built around one rig per NINA installation.

Filter offsets

Filters in a wheel are rarely perfectly parfocal: after you focus through one filter, another sits a few hundred focuser steps away from its best focus. NINA can compensate. Give each filter a **focus offset**, turn on **Use filter offsets**, and NINA moves the focuser by the difference between the two filters' offsets on every filter change, with no autofocus.

The hard part is getting the numbers. NightFront measures them for you, during twilight or any idle slot, writes them into NINA's own offset table, and never stops the night to ask you to accept them. This page covers the sequence instruction that measures, the Options page that shows what it found, and the few NINA settings you set once by hand.

Kept up to date, not calibrated once

Filter offsets are usually set once. You run a calibration, by hand or with a tool such as Filter Offset Calculator, and get one number per filter. You accept it, and it stays in NINA's table until you calibrate again. NightFront keeps measuring instead. Each clear twilight it measures a filter or two, it also records the autofocus runs your sequence makes anyway, and it keeps the table up to date by itself. You can turn it off at any time (see [Stopping or undoing](#)). NINA keeps the last values, so the worst case is the same table a one-time calibration would have left you.

	A one-time calibration	NightFront
When it measures	Once, when you run it	A bracket or two each clear twilight, for as long as you leave it on
What a value rests on	One evening's runs	The median of up to 8 brackets from the last 90 days, each from its own evening and corrected for focuser drift
Who commits it	You, after looking at the numbers	NightFront, at a safe moment, without stopping the sequence
When it changes	When you calibrate again	When 3 brackets in a row agree on a new value; once the focus tolerance is known, never for a change too small to show in your stars
Undo	Whatever you wrote down	Every write is in the history, with Restore

"Kept up to date" does not mean the table moves every night. The rules under [When an offset is written](#) are there to stop that, and a settled table usually stays as it is for weeks.

Why keeping it up to date is better

- **One evening is a small sample.** Every autofocus is noisy, and so is the offset worked out from it: a single bracket typically lands about a sixth of an autofocus step from the true value. A one-time calibration carries that evening's seeing, temperature and luck into every night after it. NightFront's offset is the median of brackets from many evenings, and the Options page shows how widely they scatter, so you can see how well each offset is known.

- **Drift is cancelled, not measured into the offset.** A focuser drifts as the tube cools, so comparing a filter's focus with a reference focused some minutes earlier adds that drift to the offset. NightFront focuses the reference before *and* after each filter and takes the reference's focus at the moment the filter was focused.
- **A bad measurement is refused rather than averaged in.** NightFront drops:
 - a bracket that spans a reconnect, runs too long or sees the temperature move,
 - a bracket whose two reference runs disagree,
 - an autofocus that failed or ended where it started.

A bracket far from the others is set aside before the median is taken, and the Options page lists it.

- **A real change is caught.** Offsets do change: a filter is reseated or replaced, the camera is refitted, the spacing or the focuser changes. After a one-time calibration the table stays wrong until soft stars give it away. Here, 3 brackets in a row that agree on a new value are written, and a single odd bracket never moves anything.
- **Offsets too small to matter are written as 0.** NightFront reads each filter's focus tolerance from your own autofocus curves, so it knows how far off focus your stars can be before anyone could see it. An offset inside that tolerance is written as 0, so NINA neither moves the focuser nor pauses guiding for that filter change.
- **Nothing holds the night.** There is no dialog to answer. A write waits for a safe moment, not for you.
- **Every write can be undone.** Each write is a line in the history with its before and after values, and **Restore** puts the old values back.

What it tests, and how

Each question below has its own section further down. In short:

1. **Is this bracket a clean measurement?** Three autofocus runs, reference, filter, reference, on one focuser connection. The three must fit within 20 minutes and 1 °C, and the two reference runs must agree within half an autofocus step ([How a measurement works](#)).
2. **Is the filter's offset known well enough to write?** At least 3 brackets, agreeing within a quarter of an autofocus step. The offset must be no more than one step from the reference and within the focuser's Max increment ([When an offset is written](#)).
3. **Is the filter parfocal with the reference?** NightFront asks this for every filter whose focus tolerance is known, with nothing to set up. The test is not "close enough": the offset, plus a margin for measurement noise, must fit inside half the filter's focus tolerance. Only then is 0 written ([The focus tolerance](#)).
4. **Are these filters parfocal with each other?** This is asked only for filters you put in the same parfocal group. A maker's claim is only a starting point: each pair's measured difference, plus the same noise margin, must fit inside the pair's tolerance. So a filter with a single bracket is never confirmed, however close it looks. A confirmed group gets one offset, and its filters share their measuring time. New brackets can withdraw a confirmation as well as grant it ([Parfocal groups](#)).
5. **Has the offset really changed?** A new value replaces a written one only when 3 brackets in a row agree on it. Once the focus tolerance is known, a change smaller than half of it is left alone, so the table does not churn by a few steps each evening.

The noise margin in 3 and 4 is 1.645 times the brackets' scatter, shrinking as brackets accumulate. That makes the test a one-sided 95 % bound: it confirms a filter is parfocal only when the measurements show it, never because they could not tell.

How a measurement works

A single autofocus result is noisy, and a focuser drifts over a night. So NightFront never compares two filters' absolute positions on their own. It **brackets** each filter between two runs through the **reference filter** (usually Luminance): reference, filter, reference. The reference's position is interpolated to the moment the filter was focused and the offset is the difference, so steady drift cancels out.

A bracket counts only when:

- all three runs completed, inside one focuser connection (a reconnect starts a new calibration, so a measurement never spans two sessions or two days),
- the three runs fit within 20 minutes and, when the focuser reports a temperature, within 1 °C,
- the two reference runs agree within **half an autofocus step** (see the limits below).

An autofocus that failed, or that ended exactly where it started, is never a measurement.

The limits scale with your focuser

Focusers differ enormously: some have a few thousand steps in all, others more than 100,000, and one step means a very different distance on each. So no limit on this page is a fixed number of steps. Each is a share of NINA's **Autofocus step size**, the spacing between the points of an autofocus curve, which you have already tuned to your focuser:

Limit	Share of the autofocus step size	Why
The two reference runs of a bracket agree	half a step	two reference runs half a step apart did not measure the same focus
The measurements' scatter	a quarter step	a quarter step of scatter is already visible defocus
A change to a committed offset, unconfirmed	a quarter step	the same reason
The size of an offset	one step	a whole autofocus step is a meaningful distance from focus, so anything beyond it is flagged for you rather than written

With an autofocus step size of 1500, for example, those are 750, 375, 375 and 1500 steps.

The focuser's **Max increment**, the largest move it accepts at once, is a hard ceiling on top: no offset larger than it is ever written. (A move between two filters is the *difference* of their offsets, so on a filter change it can still be up to twice that.)

Where NINA keeps the two values:

- **Autofocus step size:** Options → Autofocus.

- **Max increment:** Equipment → Focuser, shown while the focuser is connected. NightFront records it each time the focuser connects, so the ceiling still applies with the focuser disconnected. Until it has been recorded once, the ceiling is not applied.

If the autofocus step size is not set (zero), nothing can be judged: every filter with enough measurements is flagged and nothing is written. Measurements already taken are judged against the autofocus step size in force now, so changing it re-judges them.

The focus tolerance

How close is close enough? Every autofocus run fits a V-curve: star size against focuser position, in your focuser's own steps. From that curve NightFront reads each filter's **focus tolerance**, τ : how far from best focus you can be before the stars are **5 % larger**. Beyond τ , your stars are 5 % bigger; at half of τ they are only about 1.3 % bigger, which no one can see. Because it is read in your focuser's steps and through your own seeing, it needs no microns, no optics and no settings.

τ moves with the seeing, so a filter's τ is the median over its newest 20 well-fitted autofocus runs (a fit that explains at least 95 % of the curve, $R^2 \geq 0.95$) from the last 90 days, all made by the same autofocus engine (NINA's own and Hocus Focus measure star size differently, so their curves are never mixed). Until a filter has 5 such runs its τ is unknown, and every rule below falls back to its quarter-step form, exactly as before. NightFront reads the curves from the autofocus reports NINA already writes; Hocus Focus's tilted and blended curve models are not used.

τ decides two things, both described below: whether a parfocal group is confirmed, and when an offset close to the reference is written as 0 or left alone. The other limits above stay shares of the autofocus step size.

When an offset is written

NightFront writes a filter's offset into NINA's table only when all of these hold:

- at least **3** bracketed measurements from the last 90 days (the newest 8 are used), and their median is the offset,
- the measurements agree with each other (their scatter is within a quarter of an autofocus step),
- the offset is no more than one autofocus step from the reference, and no more than the focuser's Max increment,
- a change from a value already committed is no more than a quarter of an autofocus step, unless 3 brackets in a row agree on the new value.

A filter that fails a gate is **flagged**, not changed. The Options page says why.

Where the filter's focus tolerance τ is known, two more rules decide **what** is written:

- **An offset that does not matter is written as 0.** When the offset, plus a margin for measurement noise (1.645 times the brackets' scatter, never less than one bracket's own noise, shrinking as brackets accumulate: the same margin as the parfocal groups below), is within **half of τ** of the reference, the stars would be at most about 1.3 % larger at 0, so 0 is written. NINA then does not move the focuser at all on a change

between that filter and the reference, and does not pause guiding for it. The measured value is still shown and kept in the history.

- **A small change is left alone.** A committed offset is rewritten only when the new estimate differs from it by more than **half of τ** , so the table does not churn by a few steps each evening. A real change smaller than that waits until it grows past it. (A 0 from the first rule is written over a committed offset that is itself within half of τ of the reference, once: after that the next 0 matches it. Over a committed offset *beyond* half of τ from the reference, an estimate within half of τ of it is left alone instead, so a table at -400 is never pulled back to 0 by one estimate near the reference, and two estimates either side of the band cannot rewrite it each time.)

Where τ is unknown, every estimate that passes the gates is written as measured, as before.

When 4 or more brackets are in use, one that sits far outside the others is **set aside** before the median is taken: more than 3 times the brackets' own scatter from their median, and never less than about half an autofocus step (3×0.175 of a step, one bracket's own noise). At least 3 are always kept. A set-aside bracket is not deleted: the Options page lists it under its filter, with its value and how far it was from the median.

All offsets are written relative to the reference filter's current value in NINA's table, so a table you set up some other way keeps its meaning. NightFront writes only each filter's focus offset. It never touches the autofocus camera offset, never turns **Use filter offsets** on or off, and never changes which filter is flagged as the autofocus filter.

A write waits for a safe moment: while the wheel is on the reference filter, or right after an autofocus. It never writes while the wheel is turning. Each write is one line in the history on the Options page, with a **Restore** button, and produces one NINA notification, so a night is never silently re-tuned.

Parfocal groups

Some filters are sold as parfocal with each other: a maker may say its broadband filters all focus alike, and its narrowband filters too. That is a claim about the filters, not about your telescope, whose optics can still bring two colors to focus at different points. So NightFront treats the claim as a starting point only, and lets your own measurements decide.

On the **Filter offsets** page, give each filter you believe is parfocal with others the same **Parfocal group** (1, 2 or 3; **none** is the default, and with no group chosen nothing changes). A group may include the reference filter. The choice is kept for each NINA profile.

A group is **confirmed** only when the measurements show every member's difference to be smaller than its **tolerance: half the focus tolerance τ** of the pair (the smaller of the two filters'), beyond which the stars of one would be more than about 1.3 % larger at the other's focus. While either filter's τ is unknown, that pair's tolerance is a quarter of an autofocus step instead. For each pair of members, the difference between their average measurements, plus a margin for measurement noise, must still fit inside the tolerance. The margin is 1.645 times the brackets' scatter (never less than one bracket's own noise, 0.175 of a step), shrinking as each filter gathers more brackets. So a filter with only one bracket is never confirmed, and neither is a pair of filters with two brackets each, however close the numbers look. In a group with the reference filter, each member is also held against zero the same way, within half of that member's own τ (a quarter step while it is unknown); there one other filter with two

brackets close to the reference can be confirmed, but nothing is committed until the group has 3 brackets in all, and the page says it is waiting for them.

- A **confirmed** group works out **one** offset for all its members: the median of all their brackets together, held to the limits above. A group that includes the reference filter commits **0** for every member. NINA's table still has one entry per filter, each holding the group's value, except that the rule above that leaves a small change alone is applied to each member against its own committed offset, so members of a group that does not include the reference can hold values that differ by less than half of τ . A member whose entry already holds that value is still recorded in the history, once, so it counts as committed on the page (and in the NightFront app).
- A group that is **not confirmed** changes nothing: each filter keeps its own offset. The page names the pair that fails and by how much, for example *SII is 190 steps from Ha, bound 670; tolerance $\frac{1}{2} \times 690 = 345$ (or tolerance 410, a quarter step, while τ is unknown).*
- A confirmed group's one offset follows the two rules of the previous section too, with the smallest τ among its members (and only when every member has one), so the group is written as 0, or left alone, as one.
- The question is asked again every time an offset is worked out, so new measurements can withdraw a confirmation as well as grant it.
- Once a group is confirmed, the *NightFront Filter Offsets* instruction measures one of its members per run (the one with the fewest recent measurements) instead of every member.

The NightFront Filter Offsets instruction

NightFront Filter Offsets is in the NightFront category of NINA's sequencer. It runs bracketed autofocus (reference, filter, reference, next filter, reference, and so on) over the filters your NightFront targets use. The filters with the fewest recent measurements go first, then the ones measured longest ago, so a first set of offsets builds up over several evenings rather than in one. A confirmed parfocal group takes one turn per run, through whichever member comes first.

Field	What it does
Budget	How long the instruction may run. 0 (the default) means the evening twilight: until the Sun reaches -18° , and never longer than the -12° to -18° window. Any other number runs for that many minutes.

Things to know:

- **Nothing waits for you.** There is no tool window and no confirmation dialog, which is the point: a sequence is never held at the end of a measurement.
- **It never waits for the Sun either.** Before -12° it is given the length of the twilight window from now, so it still ends by -18° . After -18° it has nothing to do and finishes at once, unless you give it a fixed budget. Put NINA's **Wait for Sun Altitude** before it if you want it to start at a particular time.
- **It will not start in daylight.** With the Sun higher than -6° (and no fixed budget) it exits quietly and says the sky is too bright to autofocus, instead of spending the window on failed runs.
- **Bracket by bracket.** It starts a bracket only when enough budget is left to finish it, and the run in progress when the budget ends is cancelled (NINA restores the focuser). A cut-off bracket is thrown away, never counted.

- **It checks the wheel arrived.** If a filter change does not reach the filter asked for, that step fails without an autofocus, so a run can never be recorded against the wrong filter.
- **It stops when it should.** Three failed autofocus runs in a row end the run, and so does a measurements file the plugin cannot write.
- **It ends on the filter your sequence images with next**, when it can tell, and otherwise on the reference filter.
- **A quiet exit is not an error.** With *Manage filter offsets* off, no reference filter set, nothing to measure, or a filter flagged as the autofocus filter while offsets are on, it finishes at once and its status line says why. NINA only flags the instruction when the camera, focuser or filter wheel is not connected.

A bracket is about 12 to 16 minutes, and roughly 1 autofocus run in 10 fails, so one twilight usually yields one to three brackets. Expect a first full set of offsets for six filters to take about a week or two of evenings, and about one bracket per filter a week to keep them fresh.

It never moves the mount

The instruction autofocuses wherever the telescope is pointing; it never slews. Put a slew to a star field in front of it: high in the sky, away from the Moon, with tracking on.

Where it goes in a sequence

Put it in the "Start of night" block of the typical sequence in [Instructions and triggers](#), after the wait for the Sun and the slew, and before Nightly Update:

```
"Start of night" (runs once)
├ ...your own startup: cool camera, unpark, open roof...
├ Wait for Sun Altitude ..... below -12°
├ Slew to a star field
├ NightFront Filter Offsets .... Budget 0 (0 = evening twilight, Sun -12° to -18°)
└ Nightly Update
```

It loops through the brackets itself, so it needs no loop container. It reads the filters in use from the NightFront config, so it does not need Nightly Update to run first. Placed before it, it ends on the reference filter, because the plan is not loaded yet.

If you run the filter offsets every night, run Nightly Update after them, as above. A plan Nightly Update generates starts at the moment it runs, or at the start of your planning window if that is later. Run after the offsets, it plans the time that is actually left. Run before them, it plans targets into the twilight the offsets then spend autofocusing, and those first targets are cut short or skipped when the NightFront Container finally starts.

What it costs in twilight

If the NightFront config's twilight type is **Nautical** (or **Civil**), NightFront can plan imaging from -12° (or -6°), so on the evenings the instruction runs, imaging starts when it finishes instead. With Nightly Update after it, the plan simply starts then. With **Astronomical** twilight the window comes before planned imaging and costs nothing. A fixed, smaller budget (15 minutes is about one filter) spends less but builds the offsets up more slowly.

A plan you exported from the desktop app (TargetsForTonight_<date>.json in the data folder) is imported as it was planned, from the start of its window, wherever Nightly Update sits. If you export plans and run the offsets every night, use **Astronomical** twilight so the plan begins after them.

The Filter offsets page

On the Options tab (**Options** → **Plugins** → **NightFront**), the **Filter offsets** page shows:

- **Manage filter offsets**, the on/off switch. It is **off** by default. Off means nothing is recorded and nothing is committed automatically. A **Restore** still works with it off, so turning it off is also how you stop and undo.
- **Reference filter**, the filter name exactly as it appears in the NINA profile's filter list. Blank means nothing is committed.
- The **limits** in force, in steps, worked out from NINA's autofocus step size, and the focuser's Max increment once it has been recorded, followed by each filter's focus tolerance τ once it is known.
- **Filters**, a table with one row per filter: its status (not enough brackets, ready, or flagged), the committed offset in steps, the latest estimate and its scatter, how many brackets stand behind it (and how many were set aside), when its last bracket was measured (an autofocus outside a bracket does not count), its **Parfocal group**, and any flags. Each set-aside bracket gets its own line under the row, and so does the focus tolerance whenever it decided what was written, for example *R is 58 steps from L, bound 333; tolerance $\frac{1}{2} \times 738 = 369$, inside it, so R commits 0.*
- One line per **parfocal group** under the table, saying whether the group is confirmed and, if not, which pair fails.
- **History**, every write with its before and after values, and a **Restore** button on each entry that reached NINA's profile.
- A **setup note** that follows from your current NINA settings (below).

Refresh, beside the **Filters** heading, re-reads the measurements file and NINA's profile. The measurements and history live in <config-name>.filter-offsets.json beside the NightFront config chosen on the Setup page. If you have not set a config there, the file is session-config.filter-offsets.json in the data folder, which the desktop app does not read, so set the config.

Stopping or undoing

Each of these is available at any time, and none of them needs the others:

- **Take the instruction out of the sequence.** Active measuring stops. While **Manage filter offsets** is on, the autofocus runs your sequence makes anyway are still recorded.
- **Turn off Manage filter offsets.** NightFront stops recording and never writes again. NINA's table keeps the last values, so your imaging is unchanged: you are left exactly where a one-time calibration would have left you.
- **Restore** a history entry to put back the values from before it. This works with **Manage filter offsets** off.
- **Turn off NINA's Use filter offsets** (Options → Equipment → Focuser). NINA then ignores the table altogether.

Turning NightFront off never undoes anything by itself. To return to the values you had before NightFront, **Restore** the earliest entry for each filter.

Set up NINA once, at the end

NightFront never changes these; the Options page tells you which still need doing. This is the **end state**, not where you start. While the offsets are still being measured, leave **Use filter offsets** off and **keep the Autofocus after filter change trigger** in your sequence template: until every filter in use has a committed offset, that trigger is what keeps each filter in focus. Do steps 1 and 3 **together**, once the Options page says every filter has an offset. Removing the trigger earlier leaves a filter change with no refocus and no offset.

1. **Use filter offsets** on (Options → Equipment → Focuser), but only once every filter in use has a committed offset. With it on and a filter missing an offset, NINA moves the focuser by the wrong amount.
2. **No filter flagged as the autofocus filter.** With offsets on and a filter flagged, NINA redirects every autofocus through that one filter. With none flagged, every autofocus runs through the filter in the wheel and the offsets carry the focus to the next.
3. **Remove the Autofocus after filter change trigger** from your sequence template, at the same time as step 1. In NINA 3.2 it ignores **Use filter offsets** and runs a fresh autofocus on every filter change, which overwrites each offset move. Leave the temperature and HFR triggers in place.

Guiding can keep running through a filter change. An offset move is a few hundred steps between exposures, so NINA's **Disable guiding on filter change** is not needed; pause guiding during autofocus as usual.

On the desktop app, **Analysis** → **Autofocus** shows the same measurements (committed offsets, last measured, flags) in a standing *Filter offsets* panel whenever the measurements file is present, and the filter-offsets suggestion beside it reads it.

A recommended arrangement

Already using offsets? See [If you already use filter offsets](#) below; steps 2 and 3 here are for an installation that starts with none.

1. Turn on **Manage filter offsets** and type your reference filter.
2. Add *NightFront Filter Offsets* to the "Start of night" block as laid out above, after your equipment is connected and cooled and a slew to a star field, leaving the budget at 0, so each clear evening measures a bracket or two. While offsets build up, leave NINA's **Use filter offsets** off and keep the **Autofocus after filter change** trigger. The table *Manage filter offsets* writes does nothing until NINA's switch is on, so your imaging is unchanged.
3. Only when the Options page says every filter has an offset, do "Set up NINA once, at the end" steps 1 to 4 above, turning **Use filter offsets** on and removing the **Autofocus after filter change** trigger in the same sitting, and leave the instruction in to keep the offsets fresh. Turning offsets on earlier would move the focuser by the wrong amount for any filter still missing one.
4. After a week or two, compare star size in each filter with the reference. If a filter looks soft, its row will usually show why; **Restore** puts back the previous value.

If you already use filter offsets

This is for an installation whose NINA offset table is already filled and in use: **Use filter offsets** is on, with values from Filter Offset Calculator or measured by hand. You do not have to turn your offsets off while NightFront builds its own. The "leave **Use filter offsets** off" part of the arrangement above is for an installation with no offsets yet.

- **Nothing happens until you turn on Manage filter offsets.** It ships off: with it off, nothing is recorded and nothing is written to NINA's table, and your offsets keep working as they are.
- **Your table counts as committed values.** A filter holding its own nonzero offset is treated as committed, relative to the reference filter's current value in NINA's table, so a reference filter with a nonzero offset is handled. A filter sitting at 0 or at the reference's own value, which NightFront never wrote, has no committed value and is measured from scratch.
- **Keep Use filter offsets on.** Image on your current values while brackets build up. Add the *NightFront Filter Offsets* instruction to the "Start of night" block as described above.
- **Your values guard the table against noise.** Once a filter has 3 or more brackets, an estimate within a quarter of the autofocus step size of your value is written. One further away is flagged as an unconfirmed step, and your value stays until 3 brackets in a row agree on the new one. Where the filter's focus tolerance τ is known, an estimate within half of τ of your value leaves your value as it is.
- **Writes are safe with offsets live.** A write waits for the wheel to sit on the reference filter or for the moment right after an autofocus through the current filter, and never happens while the wheel is moving. NightFront never turns **Use filter offsets** on or off and never changes the autofocus-filter flag. A changed offset is used by NINA from the next filter change.
- **Undo is always there.** Every write is a line in the Options page history with its before and after values. **Restore** puts back your original value, and works even after you turn **Manage filter offsets** off again.

Three things you do change:

1. **Unflag the autofocus filter**, if one is flagged. With **Use filter offsets** on and a filter flagged, NINA sends every autofocus through that one filter, so no other filter could be measured: the instruction exits at once and its status line says to unflag it. The Options page's setup note names the filter.
2. **Remove the Autofocus after filter change trigger** from your sequence template, if it is still there, as in "Set up NINA once, at the end" above. The Options page's setup note says the same while offsets are on.
3. **Keep one writer.** We suggest taking Filter Offset Calculator's instruction out of the sequence: both it and NightFront write the same table, and NightFront does not detect the other one.

Calibration flats in NINA

As NightFront images, the plugin quietly records what your flats will need to match: which filter was used, at what rotator angle, gain, offset and readout mode. A set of calibration instructions then works through that list and shoots matching flats, on a cloudy night or at the end of a session, without re-shooting what's already done.

Palette name	Kind	What it does
NightFront Sky Flats	Instruction	Rotates to the next outstanding requirement and shoots twilight sky flats for it
NightFront Trained Flats	Instruction	Rotates to the next outstanding requirement and shoots trained flats for it
NightFront Rotate to Next Angle	Instruction	Moves the rotator to the next requirement's angle without using it up
NightFront While Flats Remain	Loop condition	Loops while any flat requirement is outstanding
NightFront While Same Rotation	Loop condition	Loops while outstanding requirements remain at the angle the loop started at

Each of these shows `Metadata file: calibration.metadata.json` in the sequencer. That file sits in your NightFront data folder, and a single copy accumulates requirements across nights.

How flat requirements accumulate

Every time an exposure in a NightFront plan finishes, the plugin records a requirement made of:

- **the filter**,
- **the measured rotator angle**, read from the rotator at that moment (the physical position a flat will actually be shot at, not the sky angle the plan asked for),
- **the gain** and **the offset** the exposure used, and
- **the readout mode**, read live from the camera at that moment.

Requirements are recorded per exposure, so a target that stops partway through its filter list (at dawn, say) records only the filters it actually shot.

A requirement already in the file isn't added twice:

- **Angles within 1° count as the same angle.**
- **Your rotator's range is respected.** NINA reports rotator positions within the range set in your profile (full 360°, half 180° or quarter 90°). Under a half range, one physical position can read as about 0° one night and about 180° the next; the plugin treats those as the same angle, so you don't get a duplicate pair and shoot the same flats twice.
- **Gain and offset must match exactly.** A different gain or offset is a different requirement.
- **An unknown readout mode matches any mode.** Requirements recorded while the camera was disconnected, or by plugin versions before readout mode was tracked, have no mode. The first exposure

that matches everything else fills the mode in rather than adding a second requirement.

Shooting the flats

NightFront Sky Flats and **NightFront Trained Flats** each take one outstanding requirement per run, then:

1. rotate to its angle, rounded to the nearest whole degree,
2. switch to its filter and apply its gain, offset and readout mode, and
3. shoot the flats with NINA's own capture: **Twilight Sky Flats** for Sky Flats, **Trained Flat Exposure** for Trained Flats.

Which requirement comes next:

- **The rotator's current angle is preferred.** If anything is outstanding at the angle the rotator already sits at, that goes first, so a session doesn't rotate away and back.
- **Then Flat Filter Order decides,** within that angle (or overall, if nothing is left at the current angle).

A readout mode, once applied, is left on the camera afterwards; your next NightFront plan sets whatever mode its own targets need.

NightFront Sky Flats fields

Field	Default
Min exposure	0 s
Max exposure	10 s
Mean target	40 %
Mean tolerance	10 %
Dither	off
Amount	20

These pass straight through to NINA's Twilight Sky Flats. The filter, gain, offset and readout mode come from the requirement.

NightFront Trained Flats fields

Field	Default
Amount	1

Exposure settings come from NINA's own trained flat settings for the filter, as with NINA's Trained Flat Exposure instruction.

When a flats run completes

- **On success**, the requirement is stamped complete with today's date. It stays in the file, which is what stops the same filter, angle, gain and offset from being added again straight away.
- **On failure or cancellation**, the requirement goes back into the queue for another attempt.
- **A set that was skipped isn't complete**. If you skip the flats while they're running, the requirement goes back into the queue, however many frames it had shot.
- **If `calibration.metadata.json` can't be written** (another program has it open, say), NightFront keeps trying for up to a minute before it gives up. If it gives up:
 - The instruction fails once, and the reason Windows gave is in NINA's log.
 - **NightFront While Flats Remain** and **NightFront While Same Rotation** end, so you get one failure notification rather than one per loop pass. They keep ending straight away until noon (the end of that observing night), until NightFront next writes the file successfully, or until NINA is restarted, whichever comes first. Then flats resume as usual.
 - A set that was already shot is **not** put back in the queue. NightFront records it as complete the next time it can write the file. If NINA is closed before that happens, see the next point.
- **A run NINA never finished is picked up again**. If NINA closes, crashes or loses power partway through a flats set, or before NightFront could record how the set went, the requirement stays marked `"Claimed": true` in the file. Starting with the next observing night (after local noon), a flats run takes it back into the queue and shoots the whole set again. NightFront doesn't count frames from a run it never saw finish. NightFront waits until noon because a second NINA using the same data folder could still be shooting that set tonight. A claim left by a plugin version older than this one is counted from the first time this version sees it, so it comes back one night later. To shoot such a set sooner, set `"Claimed"` back to `false` by hand while NINA is closed. That still works.
- **Completed requirements expire after Flat Refresh Days** (default 30). Nightly Update drops any requirement completed longer ago than that. If you image that same combination again afterwards, it becomes an outstanding requirement once more, so your flats library refreshes on the schedule you set.
- **A filter no longer in your filter wheel fails the run**, naming the filter.

Rotating without shooting

NightFront Rotate to Next Angle moves the rotator to the angle of the next outstanding requirement (by Flat Filter Order), rounded to the nearest whole degree. It only looks; it doesn't claim or complete anything. Use it to position the rotator before a same-rotation loop, or to rotate and then run your own flats instructions.

Loop conditions for flats

- **NightFront While Flats Remain** loops while at least one requirement is outstanding, anywhere.
- **NightFront While Same Rotation** takes the angle of the next outstanding requirement on its first check, then keeps looping only while requirements remain *at that angle*. It resets each time its container starts over, so an outer loop can move it on to the next angle.

Both stop straight away if there's no metadata file yet.

Example flats sequence

This nesting shoots every outstanding requirement, finishing all of one angle before rotating to the next, so each angle is visited once:

```
"Flats" (loop)          ← condition: NightFront While Flats Remain
├─ NightFront Rotate to Next Angle
├─ "This angle" (loop)   ← condition: NightFront While Same Rotation
│   └─ NightFront Sky Flats (or NightFront Trained Flats)
```

With requirements for L and OIII at 180° and B and Ha at 90°, and a **Flat Filter Order** of L, B, Ha, OIII, this rotates to 180° and shoots L then OIII, then rotates to 90° and shoots B then Ha: two rotations rather than four.

When there's nothing left to shoot

The flats items treat an empty queue differently, on purpose.

- **NightFront Rotate to Next Angle exits quietly.** With nothing outstanding it leaves the rotator where it is and the sequence carries on. There's no "next" angle, and moving anywhere would be inventing one. An empty queue is simply the normal end of a flats session.
- **The loop conditions just stop looping.**
- **NightFront Sky Flats and NightFront Trained Flats fail.** With nothing outstanding, the sequencer shows *No calibration requirements remain in the NightFront calibration-metadata file.* and the instruction fails if it runs.

Why the flats instructions fail: skipping a rotation is harmless, but silently skipping the flats themselves is not. If a flats instruction quietly did nothing, the sequence would report success having shot no flats, and you'd only discover the missing calibration frames days later when you tried to calibrate. A failure you can see at the time, and that your failure notifications can report, is the better outcome.

In the example above the loop conditions are checked before each flats run, so the flats instruction only runs while something is outstanding at that angle, and the failure doesn't arise in normal use. You'll see it if you place a flats instruction on its own and there's nothing to do.

Note: Before your first NightFront night, there's no `calibration.metadata.json` in the data folder. The flats instructions then show *No calibration metadata file (calibration.metadata.json) exists yet in the NightFront data folder.* in the sequencer, and NINA lists it among the issues it asks you to confirm when you start the sequence. If you start anyway, this case does **not** fail: NightFront Sky Flats and NightFront Trained Flats shoot nothing, show the same message as an information notification, and the sequence carries on. The loop conditions stop straight away, and Rotate to Next Angle does nothing.

Flat settings on the Options tab

Two settings on the Options tab's Setup page drive these instructions (see [The Options tab](#)):

Setting	Default	What it does
Flat Filter Order	blank	Comma-separated filter names, for example L, B, G, R, OIII, Ha, SII . Flats are shot in this order, whatever order the requirements were recorded in. Filters you don't list come after those you do. Leave it blank to use the order requirements were recorded.
Flat Refresh Days	30	Days after a requirement's flats were completed before it's dropped so it can be reshot.

A common choice is broadband filters first, since they need the brighter twilight sky, then narrowband.