LINKTOOL USER GUIDE

OPERATIONAL GUIDE

VIRPIL CONTROLS LINKTOOL

Daily operation, simulator integration setup, profile management, rule editing, telemetry diagnostics, and recovery workflow.

Version 3.0 — Windows desktop application for VIRPIL Controls devices: LED profiles, simulator telemetry, input events, and Shift Link routing.

TELEMETRY	Source setup and validation
PROFILES	Create, open, save, and test rule sets
RULES	Build, troubleshoot, and verify LED logic
DIAGNOSTICS	Isolate device, runtime, and simulator issues

TABLE OF CONTENTS SECTI

This table of contents contains clickable links that jump to each section inside the PDF.

Primary sections are listed first, followed by operational subsections for faster navigation.

TABLE OF CONTENTS

WHAT LINKTOOL DOES

- VERSION 3 — FIRST RELEASE NOTICE

HOW LINKTOOL WORKS AT A HIGH LEVEL

BEFORE YOU START

- INSTALLATION AND UPDATES
- USEFUL LINKS

QUICK START

FIRST LAUNCH: WHAT TO DO ON THE VERY FIRST SCREEN

INTERFACE OVERVIEW

- WINDOW HEADER
- SIDEBAR NAVIGATION
- 1. DASHBOARD
- 2. TELEMETRY DRAWER AND TELEMETRY LAB
- 3. TRANSFORMS PAGE
- 4. SETTINGS
- 5. DIAGNOSTICS

WORKING WITH TELEMETRY TRANSFORMS

- MSFS NOTE
- SAVING AND BACKING UP TRANSFORMS

LANGUAGE, THEME, DOCUMENTATION, AND BUG REPORTING

- CHANGING THE LANGUAGE
- CHANGING THE THEME
- OPENING THE BUILT-IN DOCUMENTATION PDF
- WHERE THE BUG REPORT WINDOW IS

DEVICE WORKFLOW

- DETECTING DEVICES
- WHAT TO VERIFY
- INPUT MONITOR IN DEVICE PREVIEW

SIMULATOR AND TELEMETRY WORKFLOW

- CHOOSING THE ACTIVE SIMULATOR
- CHECKING TELEMETRY
- TELEMETRY LAB WORKFLOW
- USING TELEMETRY WHILE BUILDING RULES
- MISSING TELEMETRY PARAMETERS

PROFILES

- BASIC PROFILE ACTIONS
- EXPORT AND IMPORT PROFILE BUNDLES
- WHEN TO USE NEW PROFILE
- AUTO-SAVE AND MANUAL SAVE

AUTOLOAD BY SIMULATOR

WHAT A PROFILE FILE ACTUALLY IS

WHERE PROFILES ARE SAVED

RULES AND THE RULE EDITOR

COMMON RULE TYPES

WHAT A RULE USUALLY CONTAINS

THE SAFEST FIRST RULE

A FAST WAY TO READ ANY RULE

PRACTICAL RULE SETUP SCENARIOS

SCENARIO 1. A SIMPLE TELEMETRY RULE FOR A STATE INDICATOR

SCENARIO 2. A BUTTON RULE FOR A CLEAR MANUAL EVENT

SCENARIO 3. AN AXIS RULE FOR THRESHOLD INDICATION

SCENARIO 4. A GRADIENT RULE FOR CONTINUOUS VALUES (TYRE OR BRAKE TEMPERATURE)

WORKING IN THE RULE EDITOR

WHAT RULE MODE MEANS

DUPLICATE RULE WIZARD

HOW RULE PRIORITY WORKS

TARGET CONFLICT BADGES

WHAT BUTTON MODE MEANS

WHAT SHIFT MEANS IN BUTTON RULES

WHILE EDITING

COMMON RULE SETUP MISTAKES

PRACTICAL ADVICE

MANUAL LED CONTROL

SETTINGS AND INTEGRATIONS

HOW TO OPEN AND USE SETTINGS CORRECTLY

WHAT EACH SETTINGS TAB IS FOR

WHAT YOU USUALLY CONFIGURE HERE

GENERAL SETTINGS: WHAT EACH IMPORTANT OPTION MEANS

WHEN TO GO TO SETTINGS

PRACTICAL INTEGRATION ORDER

INTEGRATION QUICK MAP

INTEGRATION SUMMARY TABLE

DCS WORLD

MICROSOFT FLIGHT SIMULATOR / MSFS 2024 / PREPAR3D V5

IL-2 GREAT BATTLES

XPLANE 11

XPLANE 12

ELITE DANGEROUS

WAR THUNDER

AUTOMOBILISTA 2

ASSETTO CORSA EVO

LE MANS ULTIMATE

IRACING

PROJECT MOTOR RACING

PDF-FRIENDLY INTEGRATION CHECKLIST

END-TO-END EXAMPLE: FROM EMPTY PROFILE TO A WORKING LED

DIAGNOSTICS AND TROUBLESHOOTING

DIAGNOSTICS TAB: WHAT IT GIVES YOU

RECOMMENDED BUG REPORT WORKFLOW

SCENARIO 1. THE DEVICE IS NOT VISIBLE

SCENARIO 2. THE DEVICE IS VISIBLE BUT THE LEDS DO NOT RESPOND

SCENARIO 3. TELEMETRY IS NOT ARRIVING

SCENARIO 4. A RULE SAVES BUT NEVER TRIGGERS

SCENARIO 5. AN IMPORTED LEGACY PROFILE LOOKS WRONG

SCENARIO 6. EXPLAIN DOES NOT WORK OR SHOWS AN ERROR

PRACTICAL TROUBLESHOOTING RECIPES BY SIMULATOR

DCS WORLD: THE DEVICE IS VISIBLE BUT TELEMETRY NEVER ARRIVES

SIMCONNECT FAMILY: MSFS/PREPAR3D IS SELECTED BUT NO PARAMETERS APPEAR

PREPAR3D V5: SIMCONNECT ERROR / VERSION_MISMATCH / EMPTY TELEMETRY

IL-2 GREAT BATTLES: THE GAME IS RUNNING BUT THE SNAPSHOT IS EMPTY

ELITE DANGEROUS: THE SIMULATOR IS SELECTED BUT NO DATA APPEARS

XPLANE 11: SETTINGS ARE FILLED IN BUT TELEMETRY IS STILL SILENT

XPLANE 12: SETTINGS ARE FILLED IN BUT TELEMETRY IS STILL SILENT

WAR THUNDER: DATA ARRIVES INCONSISTENTLY

AUTOMOBILISTA 2: NO SHARED MEMORY DATA

ASSETTO CORSA EVO: NO SHARED MEMORY DATA

LE MANS ULTIMATE: NO SHARED MEMORY DATA

IRACING: NO SHARED MEMORY DATA

PROJECT MOTOR RACING: NO UDP TELEMETRY

UNIVERSAL ISOLATION ORDER

FAQ

DO I ALWAYS NEED A NEW PROFILE

WHICH IS BETTER: AUTO-SAVE OR MANUAL SAVE

WHY SHOULD I TROUBLESHOOT ONE SIMULATOR AT A TIME

WHY IS IT BETTER TO TEST ONE RULE FIRST

CAN I TEST THE DEVICE BEFORE USING A SIMULATOR

DOES MY CHATGPT OR CLAUDE PHONE SUBSCRIPTION WORK AS AN API KEY

X-PLANE 11 OR X-PLANE 12 — WHICH ONE DO I SELECT

WHERE DO I CHANGE THE LANGUAGE AND THEME

WHERE IS THE BUG REPORT WINDOW

WHERE IS THE PDF MANUAL IN THE APP

WHY DO ALL LEDS TURN OFF OR CHANGE COLOR WHEN I PRESS START

WHY DOES RESET LEDS BEFORE START SEEM TO DO NOTHING

WHY CAN IL-2 GREAT BATTLES LOOK BROKEN EVEN WHEN THE RULE IS FINE

SCREENSHOT PLACEHOLDERS FOR FUTURE PDF WORK

RECOMMENDED WORKING STYLE

WHAT LINKTOOL DOES

SECTION 01

LinkTool connects simulator telemetry, input events, and lighting rules to LED behavior on supported VIRPIL Controls devices. Version 3 is a full rewrite of the legacy VPC Link Tool with a modern WPF interface, a testable rule engine, and expanded simulator integrations.

WHAT LINKTOOL DOES

You can use it to:

- detect connected VIRPIL devices;
- select the active simulator and inspect live telemetry parameters;
- open an existing `.led.json` profile or create a new one from scratch;
- configure lighting rules based on telemetry, buttons, axes, and keyboard shortcuts;
- test individual LEDs or all LEDs manually;
- inspect diagnostics when a rule does not trigger or a device does not respond.

VERSION 3 — FIRST RELEASE NOTICE

This is the **first public release of LinkTool v3**. Core workflows are in place, but some areas may still be rough around the edges:

- individual simulator integrations may need extra setup or tuning;
- legacy profile import may require manual review of device mapping and parameters;
- UI polish, diagnostics, and edge-case behavior are still evolving.

If something does not work as expected, use **Report issue** in the app header or collect logs from **Settings** → **Diagnostics**. Your feedback helps stabilize v3 faster.

SECTION 02

At a high level, LinkTool repeats the same cycle while it is running:

HOW LINKTOOL WORKS AT A HIGH LEVEL

1. read data from the active simulator or local input sources;
2. update the current telemetry and input snapshot;
3. evaluate rules from the working ruleset;
4. apply the resulting LED states to connected devices.

This is useful to remember during troubleshooting. If the LED result is wrong, the problem usually lives in one of these four stages.

BEFORE YOU START

SECTION 03

Before working in LinkTool, check a few basics:

BEFORE YOU START

- your VIRPIL devices are connected and visible to the operating system;
- the simulator you want to use is running or ready to run;
- if you plan to use telemetry, the relevant integration is configured in Settings;
- if you want to continue an existing setup, you have access to the `.led.json` profile file you need.

If this is your first time using the application, the best starting point is a new empty profile with one simple rule. That keeps the behavior predictable and makes troubleshooting much easier.

INSTALLATION AND UPDATES

LinkTool is distributed as a Windows desktop application. Current builds and release notes are published on software.virpil.com.

Typical installation flow:

1. Download the latest stable LinkTool build.
2. Run the installer and choose the installation directory.
3. Start LinkTool and refresh the device list.
4. Before your first simulator integration, open **Settings → Integrations** and configure only one simulator first.

If the application becomes unstable after repeated experiments, use the same conservative recovery path described in the legacy manuals:

1. keep backups of your working `.led.json` profiles;
2. reinstall the latest stable release if needed;
3. redeploy the DCS or X-Plane 11 integration files if you use those simulators; for **X-Plane 12**, Integrations and the Datarefs page are enough;
4. reopen a simple test profile and validate the chain `device → telemetry → one rule`.

USEFUL LINKS

- VIRPIL Controls
- software.virpil.com
- VIRPIL Controls Forum — LinkTool
- DCS integration script (GitHub)
- X-Plane FlyWithLua (GitHub)
- X-Plane LinkTool script (GitHub)
- X-Plane Datarefs

QUICK START

SECTION 04

Use this as the shortest reliable getting-started sequence:

QUICK START

1. Start LinkTool.
2. On the main screen, click `Refresh devices` if the device list has not refreshed automatically yet.
3. Make sure your device appears in the devices area.
4. Select the active simulator.
5. Open the telemetry panel or **Telemetry lab** and verify that parameters are arriving.
6. Click `New profile` to create a clean profile.
7. Add a new rule.
8. Set the target device, parameter or event source, LED color, and LED mode.
9. Save the profile.
10. Start LinkTool if rule processing is not active yet.

If you do not want to begin with rules immediately, open the manual LED control area first and confirm that the device responds to direct lighting commands.

> Screenshot placeholder SC-01: Main dashboard after startup with connected devices and key actions.

SECTION 05

For a new user, the first launch can feel deceptively simple: the window opens, but it is not always obvious which controls matter immediately and which ones can wait.

FIRST LAUNCH: WHAT TO DO ON THE VERY FIRST SCREEN

Use this simple order:

- first confirm that LinkTool sees the hardware;
- then confirm that LinkTool sees the data source;
- only after that start building rules.

The first things worth checking are:

1. The device list. If your VIRPIL device is not visible here, rule work can wait.
2. The active simulator selector. Make sure it matches the simulator you are actually preparing to test.
3. Telemetry tools. If you want rules to react to simulator data, make sure values are coming in before you edit the rule. Start with the compact **Telemetry snapshot** drawer on the Dashboard; open **Telemetry lab** when you need search, watchlists, or a larger parameter list.
4. The profile actions. Decide whether you are opening an existing `.led.json` profile or starting with `New profile`.

If you skip these four checks and jump directly into editing rules, it becomes much harder to tell whether a failure is caused by USB, simulator integration, or the rule itself.

INTERFACE OVERVIEW

SECTION 06

Most day-to-day work happens in a few core areas of the application. The important part is not only what each area is called, but what you are supposed to do there.

INTERFACE OVERVIEW

WINDOW HEADER

The top header contains quick controls that affect the whole application rather than one single rule.

From left to right, the most important header controls are:

- the language selector, which switches the interface language;
- the appearance button, which toggles the current theme;
- the `Report issue` button, which opens the bug-report dialog;
- the documentation button, which opens the bundled PDF manual in the current language.

Treat the header as the place for quick application-wide actions. If you want to change how the whole application looks or report a problem, start there.

Important practical detail:

- language and theme affect the whole application, not one profile;
- the documentation button opens the packaged PDF from the application output, so you can use the same manual that ships with the app.

SIDEBAR NAVIGATION

The sidebar is the main page switcher. It is how you move between the Dashboard, the dedicated `Transforms` page, Settings, diagnostics pages, and simulator pages such as SimVars when available.

If you are looking for a control and do not see it on the Dashboard, the next place to check is usually the sidebar entry for `Settings`.

1. DASHBOARD

The dashboard combines most of the everyday workflow:

- connected device list;
- simulator selection;
- profile actions;
- rules list;
- manual LED controls;
- status and system messages.

If you use LinkTool regularly, this is where you will spend most of your time.

In practice, the Dashboard is where three kinds of work happen:

- everyday control: select the simulator, start or stop LinkTool, watch status;
- profile work: create, open, save, and edit rule sets;
- hardware verification: test LEDs manually and confirm that the selected device reacts.

If you are unsure where to begin, begin on the Dashboard. It is the shortest route to understanding whether the app, the device, and the simulator are all connected correctly.

2. TELEMETRY DRAWER AND TELEMETRY LAB

LinkTool gives you two complementary ways to inspect live simulator parameters.

TELEMETRY SNAPSHOT DRAWER

The telemetry drawer opens on the Dashboard and shows a compact live snapshot of current parameters. It is especially useful when:

- you need to verify that simulator data is arriving at all;
- you need the exact parameter name and current value before building a rule;
- you want a quick check without leaving the main window.

For many users, this drawer becomes the fastest validation tool in the application. It answers the question: "Does LinkTool actually see the simulator state I want to react to?"

Use the drawer **search field** to filter the live parameter list by name, display label, or current value when the snapshot grows too long to scan manually.

When a **racing simulator** is selected (**Automobilista 2**, **Assetto Corsa Evo**, **Le Mans Ultimate**, **iRacing**, or **Project Motor Racing**), the drawer groups parameters under section headers such as `Tyre`, `Brake`, `Fuel`, `Pit`, `Flag`, and `Session` instead of a single flat list. Flight simulators keep the flat list. The search field still filters across all groups.

If the answer is no, rule editing is premature. Fix telemetry first.

The drawer also contains an **Open telemetry lab** action when you need the full explorer.

> **Performance note.** With LinkTool running and live telemetry active, keeping the **Telemetry snapshot** drawer open while using a very low shared-memory poll interval (for example 10–20 ms) can cause brief UI stutter on some PCs. The default **50 ms** poll interval is usually fine. If the app feels sluggish with the drawer open, try **100 ms** under **Settings → Integrations → Shared Memory**, or close the drawer while driving. LinkTool throttles drawer updates automatically, but high-frequency sim telemetry plus a live parameter list still uses CPU.

TELEMETRY LAB

Telemetry lab is a separate resizable window for deep telemetry work. Open it from the green telemetry icon on the **Dashboard top bar**, or from the drawer's **Open telemetry lab** button.

Use Telemetry lab when:

- the simulator exposes a large parameter set and the compact drawer is hard to navigate;
- you are testing rules and need search, grouping, deltas, or a personal watchlist;
- you want to jump from a parameter directly into rule editing;
- a parameter name is unclear and you want a short plain-language explanation before writing a rule.

What Telemetry lab provides:

- **Filter tabs:** `All`, `Watchlist`, `Changed`, and `In rules`.
- **Group chips:** quick filtering by parameter family such as `Tyre`, `Brake`, or `Fuel`.
- **Search field:** filter by name, value, or tokens such as `changed:`, `used:`, `watched:`, `pinned:`, and `group:Tyre`.
- **Watchlist:** click the **eye** icon on a row to watch a parameter for the current simulator. Watchlists are saved per simulator in application settings.
- **Create rule:** click the **plus** icon on a row to create a new telemetry rule with that parameter already filled in. LinkTool opens the Dashboard rule editor and pauses LinkTool while the editor is expanded, same as for any other rule edit.
- **Explain parameter:** click the **EXPLAIN** icon (question/comment) on a row to ask the configured AI provider what the parameter means in the current simulator. Useful when a name is cryptic or you are unsure how to use it in a rule.
- **Copy name:** double-click a row to copy the parameter name to the clipboard.
- **Delta column:** shows numeric change since the previous telemetry sample when the value is numeric.
- **Freeze snapshot:** pause live updates to inspect one moment in time.
- **Keep window on top:** useful when the lab is docked beside the main window during testing.

EXPLAIN button

Each parameter row includes an **EXPLAIN** button (question/comment icon) at the right side of the row actions.

Use it when you see a telemetry name such as `AoA`, `NWS`, or `SIM ON GROUND` and want a quick answer to questions like:

- what does this value represent in the **currently selected simulator**;
- what units or range are typical;
- how might it be used in a simple LED rule.

When you click **EXPLAIN**, LinkTool opens a small dialog, sends a short request to your **active AI provider**, and shows a concise answer in your **selected UI language**. If LinkTool already has a live value for that row, the explanation can reference it.

Before EXPLAIN works for the first time:

1. open **Settings → AI Integrations**;
2. enable **Enable telemetry parameter explanations**;
3. choose **Active provider** (Groq is the free default);
4. paste an API key on the matching sub-tab;
5. click **Save settings**.

EXPLAIN does not edit your profile or create rules. It is optional reference help while you browse telemetry. Repeat requests for the same parameter may show **Cached answer** immediately — LinkTool stores prior explanations locally to reduce API usage. Full provider and billing details are in the **AI Integrations** section under Settings.

Telemetry lab remembers its window size, position, always-on-top state, and per-simulator watchlists between sessions.

> **Performance note.** Telemetry lab is optimized for high-frequency telemetry, but simulators with very fast shared-memory updates can still load the UI on low-end PCs. If the lab feels heavy during gameplay, use **Freeze snapshot**, raise the shared-memory poll interval under **Settings → Integrations → Shared Memory**, or close the lab while driving and reopen it during setup or pit stops.

3. TRANSFORMS PAGE

The `Transforms` page is the single place where LinkTool now creates telemetry aliases and numeric normalization for every simulator.

Use it when:

- raw telemetry names are too technical or inconvenient for rules;
- a simulator reports a value in the wrong scale for the rule you want to write;
- you want one clean alias name that is easier to reuse across multiple rules;
- you want to back up or restore your transform definitions separately from profiles.

Important practical rule:

- raw telemetry parameters stay unchanged;
- a transform publishes an additional telemetry name for the selected simulator;
- rules can use either the raw name or the transformed alias name;
- transforms only take effect after `Save settings`.

Examples of what belongs here:

- convert a raw `0.87` gear value into `87` by multiplying by `100` and rounding;
- rename a hard-to-read telemetry key into a short alias such as `Gear Percent`;
- normalize one simulator-specific parameter so it is easier to use in rule thresholds.

4. SETTINGS

Settings contains general application options, simulator connection settings, update options, diagnostics helpers, and other supporting options.

The fastest way to think about Settings is this:

- `General` defines how LinkTool behaves as an application;
- `Integrations` defines where simulator data comes from and where support files are copied;
- `AI Integrations` configures optional remote AI providers for telemetry **EXPLAIN** help;
- `Updates` handles update workflow;
- `Diagnostics` helps you inspect logs and collect information when something goes wrong.

5. DIAGNOSTICS

Use diagnostics when something behaves unexpectedly: a rule does not fire, the device does not respond, telemetry is missing, or a profile behaves differently than expected.

Diagnostics is not only for advanced users. It is the correct place to go when you want a concrete answer to questions like:

- where are the logs stored;
- what log level is active;
- what is the latest log file;
- how do I export logs for support or a bug report.

SECTION 07

The main idea is simple: transforms are optional helper layers on top of raw simulator telemetry. They do not replace the source value. They publish an extra name that is usually easier to read or easier to compare inside rules.

WORKING WITH TELEMETRY TRANSFORMS

Use this workflow:

1. Select the simulator you care about.
2. Open the telemetry drawer or Telemetry lab and confirm the raw source parameter is arriving.
3. Open the `Transforms` page from the left sidebar.
4. Click `Add transform`.
5. Choose the simulator, enter the raw source parameter name, set the output name, and choose the numeric operation, factor, and rounding if needed.
6. Click `Save settings`.
7. Return to the Dashboard rule editor and use the new alias in your rule.

What each field means:

- `Simulator`: the transform only applies to that simulator.
- `Source parameter`: the raw telemetry name LinkTool already receives.
- `Output name`: the extra alias name that rules will see.
- `Transform`: the numeric operation such as multiply or divide.
- `Factor`: the numeric multiplier or divisor.
- `Rounding`: optional floor, ceiling, round, or truncate behavior after the math step.

If `Output name` is left blank, LinkTool automatically uses a fallback name based on the source parameter, for example `<source> (Transformed)`.

MSFS NOTE

The **SimConnect Vars** page defines raw inputs for the SimConnect family: MSFS 2020, MSFS 2024, and Prepar3D v5.

The UI is split into two columns:

- **SimVars** — standard SDK variables such as `AIRSPEED TRUE` and `SIM ON GROUND`;
- **LVars** — aircraft/add-on local variables such as FBW A320 or Working Title CJ4 states.

Use the page **search field** to filter both columns by variable name, display name, or live value. LinkTool also validates enabled definitions in the background and flags duplicate names before you save.

For each entry you configure:

- **Source** — `SimVar` or `LVar`;
- **Display name** — friendly label in the UI;

- **Variable name** — SDK name for SimVars; bare name without the `L:` prefix for LVars;
- **Type** — SimConnect data type;
- **Units** — measurement units;
- **Epsilon** — minimum change threshold used to ignore tiny fluctuations (see the `i` hint in the editor).

The **Add SimVar** and **Add LVar** buttons insert a new entry at the top of the matching column.

LVars work only in **MSFS 2020 (Sim Update 12+)** and **MSFS 2024**. On Prepar3D they remain in the saved config but are skipped at runtime. LVar names depend on the loaded aircraft; discover them in MSFS Dev Mode via **Tools → Behaviors → Local Variables**.

If you want an alias or normalized value for MSFS, create it on the shared **Transforms** page, exactly the same way you would for DCS, IL-2, Elite Dangerous, X-Plane, War Thunder, Automobilista 2, Assetto Corsa Evo, Le Mans Ultimate, iRacing, or Project Motor Racing.

The **SimConnect Vars** toolbar provides separate actions for backup and migration:

ACTION	USE WHEN
Load preset pack	You want to merge a bundled SimVar or LVar starter set into your deck without hand-typing every entry.
Import	You have a LinkTool v3 SimVar backup — a JSON **array** of variable definitions exported from this app.
Import legacy Link Tool vars	You have an old VPC Link Tool file such as `msfs2020.json` with a top-level `"variables"` list.
Export	You want to back up the current SimConnect variable list from LinkTool v3.
Save SimVars	You want to persist the working list into application settings.
Reset defaults	You want to replace the working list with the built-in starter set.

LOADING BUNDLED PRESET PACKS

LinkTool ships curated starter variable sets for MSFS and related SimConnect simulators.

1. Open **SimConnect Vars**.
2. Click **Load preset pack**.
3. In the picker, choose a pack — for example **Generic cockpit**, **A320 cockpit**, or **Common LVars** (MSFS 2020 SU12+ and MSFS 2024 only).
4. Click **Load pack**. LinkTool merges new rows into the matching **SimVars** or **LVars** column. Entries that already match the same display name or variable name are skipped.
5. Review the status message for added and skipped counts, then click **Save SimVars**.

LVar packs appear only when your active SimConnect profile supports LVars. On **Prepar3D v5**, LVar packs are hidden because LVars are not polled at runtime.

Preset packs are curated starters, not a complete list for every aircraft add-on. Verify variable names against your loaded aircraft before building rules.

IMPORTING A LINKTOOL V3 BACKUP

1. Open **SimConnect Vars**.
2. Click **Import** and select the backup JSON file.
3. LinkTool validates the file, loads the definitions, and saves them when validation passes.

If the imported backup still contains inline transform settings from an older LinkTool build, LinkTool migrates those transforms into the shared **Transforms** workflow. Names that already start with `L:` are automatically treated as LVars.

IMPORTING LEGACY LINK TOOL VARS (`MSFS2020.JSON`)

Older VPC Link Tool builds stored MSFS variables in a different JSON shape:

```
{
  "library": "MSFS-2020-SimConnect.dll",
  "variables": [
    {
      "variable": "LIGHT LANDING",
      "type": "SIMCONNECT_DATAType_FLOAT64",
      "units": "bool",
      "length": 8
    },
    {
      "variable": "L:H160_SDK_ENG_1_N2",
      "type": "SIMCONNECT_DATAType_FLOAT64",
      "units": "percent",
      "length": 8
    }
  ]
}
```

To import that format:

1. Open **SimConnect Vars**.
2. Click **Import legacy Link Tool vars** (the second import icon on the toolbar).
3. Select your legacy JSON file. LinkTool validates the JSON and rejects files that are not legacy Link Tool vars documents.
4. Review the imported entries in the **SimVars** and **LVars** columns.
5. Click **Save SimVars** to persist the new configuration.

Important differences from regular **Import**:

- Legacy import loads the list into the editor only. It does **not** write settings until you click **Save SimVars**.
- Variables whose names start with `L:` are converted into **LVars**; the `L:` prefix is removed from the stored name.
- Standard SDK names remain **SimVars**.
- Duplicate legacy entries are skipped and reported as import warnings.
- If you select a LinkTool v3 backup by mistake, LinkTool tells you to use regular **Import** instead.

SAVING AND BACKING UP TRANSFORMS

Transforms are stored in application settings, not inside `.led.json` rule profiles.

That means:

- `Save settings` persists the working transform list on the local machine;
- `Load...` on the `Transforms` page imports a transform backup JSON file;
- `Save as...` on the `Transforms` page exports the current transform list to a backup JSON file.

This is useful when you reinstall Windows, move to another PC, or want to keep a known-good set of simulator aliases separately from your LED rule profiles.

> Screenshot placeholder SC-02: Dashboard with the main work areas highlighted.

SECTION 08

These features are easy to overlook, but they matter in day-to-day use because they affect the entire application rather than one profile.

LANGUAGE, THEME, DOCUMENTATION, AND BUG REPORTING

CHANGING THE LANGUAGE

The language selector is located in the top header of the main window.

Use it when:

- you want to switch the entire interface between available languages;
- you want the bundled PDF documentation button to open the matching localized manual;
- you are preparing screenshots or support material in a specific language.

What changes when you switch language:

- navigation labels;
- settings captions;
- status messages;
- dialog text;
- the language used by the in-app documentation button.

What does not change:

- your rules;
- your profile content;
- your simulator addresses, ports, and file paths;
- your saved profile files.

In other words, language changes the interface, not the logic.

CHANGING THE THEME

The theme toggle is also in the top header. It switches the application between the available visual themes.

This is a visual preference only. It does not affect:

- telemetry;
- rule execution;
- profile format;
- device detection;
- file locations used by simulator support files.

Use it when you want the application to be more comfortable for your lighting conditions or personal preference.

OPENING THE BUILT-IN DOCUMENTATION PDF

The `PDF` button in the top header opens the bundled documentation file that ships with the application.

How language selection works:

- when the UI language is Russian, LinkTool prefers the Russian PDF;
- when the UI language is not Russian, LinkTool prefers the English PDF;
- if the preferred PDF is missing, LinkTool falls back to the other available language.

This makes the PDF button useful not only as a convenience feature, but also as a quick confirmation that the packaged documentation is present in the current build.

WHERE THE BUG REPORT WINDOW IS

The `Report issue` button is located in the top header next to the other global controls.

Use it when:

- you encountered a reproducible bug;
- the application fails in a way you cannot diagnose from the UI alone;
- you need to send a support ticket or bug report with context.

Practical recommendation:

1. reproduce the problem once;
2. note the exact simulator, device, and profile involved;
3. open Diagnostics and collect logs if necessary;
4. then use `Report issue`.

DEVICE WORKFLOW

SECTION 09

Device Workflow guidance and operational notes for VIRPIL Controls LinkTool.

DEVICE WORKFLOW

DETECTING DEVICES

When LinkTool starts, it tries to detect connected devices automatically. If a device does not appear or was reconnected after startup, use `Refresh devices`.

The normal flow is simple:

1. Connect the device.
2. Wait for it to appear in the list.
3. If it does not appear, click `Refresh devices`.
4. Confirm that the displayed device title looks correct.

WHAT TO VERIFY

- the device is visible inside LinkTool, not only in Windows;
- the correct device is selected for the rule you are editing;
- if the device has multiple LEDs, you understand which LED or LED group the rule targets.

If you imported an older profile, it is worth checking that the target devices in the imported rules still match the real connected devices. LinkTool tries to resolve older naming patterns, but a quick manual review is still the safe approach.

INPUT MONITOR IN DEVICE PREVIEW

When **Device preview** is open, the **Button input** panel shows a live grid of pressed buttons for the monitored device.

Use it when:

- you need to discover the button number for a physical control;
- you want to confirm which input index fires when you move a switch or press a button;
- you are building a **Button** rule and want a faster path than guessing button numbers.

How to use it:

- **Hover** a cell to see the control name when LinkTool can resolve it from device metadata.
- **Single-click** a cell while a **Button** rule editor is expanded on the Dashboard — LinkTool assigns that button number to the open rule.
- **Double-click** a cell to create a new button rule for that input (LinkTool opens the Dashboard editor).
- **Right-click** a cell for **Create rule for button N**.

When a button rule editor is expanded, the monitor prefers the rule's **source device** if that device is connected. Otherwise it follows the device selected in device preview.

Screenshot placeholder SC-03A: Device preview with button input monitor and pressed-button highlights.

SIMULATOR AND TELEMETRY WORKFLOW

CHOOSING THE ACTIVE SIMULATOR

Use the dashboard to select the simulator you are currently working with. LinkTool uses the active source, so it is important that the selected simulator matches the one you are actually testing.

If **Settings → General → Autoload LED profiles** is enabled, changing the active simulator also loads that simulator's last LED profile automatically.

Practical advice: when you are validating telemetry and rules, stay on one simulator until the workflow is stable. Changing sources too early makes troubleshooting harder.

CHECKING TELEMETRY

Open the telemetry drawer or Telemetry lab and inspect the parameter list.

At this point, answer two questions:

1. Is data arriving at all?
2. Can you see the parameter you want to use in your rule?

Use the drawer for a quick yes/no check. Use Telemetry lab when you need search, grouping, watchlists, or numeric deltas while exploring a large parameter set.

If the list is empty, the problem is usually one of these:

- the simulator is not running;
- the wrong simulator is selected;
- the integration has not been configured in Settings yet;
- the telemetry source is unavailable.

TELEMETRY LAB WORKFLOW

Recommended workflow when building or debugging telemetry rules:

1. Select the correct simulator on the Dashboard.
2. Start LinkTool if you want live values while testing.
3. Open **Telemetry lab** from the Dashboard top bar.
4. Use **Search** or **Groups** to find the parameter you care about.
5. Optional: add it to the **Watchlist** with the eye icon so it stays easy to find for that simulator.
6. Confirm the live **Value** and, for numeric parameters, the **Delta** column while you change the sim state.
7. Optional: click **EXPLAIN** on the row if you need a plain-language description of the parameter before writing a rule.

8. Either double-click the row to copy the parameter name, or click the **plus** icon to create a new telemetry rule with that parameter already selected.
9. Finish the condition, LED target, and colors in the Dashboard rule editor, then save the profile.

Search tokens worth remembering:

TOKEN	MEANING
`changed:`	parameters whose value changed recently
`used:`	parameters already referenced by a rule in the current profile
`watched:` or `pinned:`	parameters on the current simulator watchlist
`group:Tyre`	parameters in the selected group prefix

Use **Freeze snapshot** when a value is changing too quickly to read. Use **Keep window on top** when Telemetry lab is parked beside the main window during a test session.

USING TELEMETRY WHILE BUILDING RULES

When you open the rule editor for a telemetry rule, use the telemetry drawer or Telemetry lab as reference data:

- find the parameter you need in the telemetry snapshot or lab list;
- confirm the exact name;
- confirm the value format;
- use **EXPLAIN** in Telemetry lab or next to the parameter field in the rule editor when you need a short AI description;
- only then set the condition in the rule.

This avoids the common case where a rule saves correctly but never triggers because the parameter name or value format is wrong.

If you already found the parameter in Telemetry lab, the fastest path is to click the row's **Create rule** button instead of typing the name manually.

MISSING TELEMETRY PARAMETERS

Not every simulator or aircraft sends every parameter. LinkTool treats a missing key as **absent**, not as zero.

CONDITION	MISSING PARAMETER
`Not equal`	does not match
`Equals`	does not match
`Greater than`, `Less than`, `In range`	do not match
`Has value`	does not match

Practical examples:

- `trimmer != 0` does **not** fire on aircraft that do not publish `trimmer`.

- To require both presence and a non-zero value, combine two conditions: `trimmer Has value` AND `trimmer != 0`.
- If you only care about positive deflection, `trimmer > 0` is often enough on its own.

The rule editor shows a short hint when `Not equal` or `Has value` is selected.

> Screenshot placeholder SC-03: Simulator selection plus the Telemetry snapshot drawer. Screenshot placeholder SC-03A: Telemetry lab with filters, groups, watchlist, and parameter table.

PROFILES SECTION

SECTION 11

In LinkTool, a profile is a set of lighting rules usually stored in a `.led.json` file.

PROFILES

BASIC PROFILE ACTIONS

The main actions are:

- `New profile` to create an empty profile;
- `New profile from template` (template icon next to profile actions) to load a bundled starter pack;
- open an existing profile;
- save the current profile;
- use `Save As` when the profile is new or must be stored under a different name;
- **Export profile bundle** (share icon) to package the saved profile for forum sharing or backup;
- **Import profile bundle** (import icon) to unpack a shared `.linkprofile.zip` into your profiles folder and open it.

EXPORT AND IMPORT PROFILE BUNDLES

Profile bundles are ZIP archives with the extension `.linkprofile.zip`. They are meant for sharing a working setup or moving a profile between PCs without hand-editing JSON.

What is inside a bundle:

- `profile.led.json` — the LED rule profile;
- `linktool-profile.json` — metadata such as simulator name, export date, and LinkTool version.

How to export:

1. Save the active profile first (`Save` or `Save As`). Export uses the saved file on disk.
2. On the Dashboard profile header, click **Export profile bundle** (share icon).
3. Choose where to save the `.linkprofile.zip` file.

How to import:

1. On the Dashboard profile header, click **Import profile bundle** (import icon).
2. Select a `.linkprofile.zip` file.
3. LinkTool unpacks the profile into its profiles `data` folder (the same area used for legacy profile storage) and opens the imported profile in the editor.

If a profile with the same filename already exists, LinkTool adds a timestamp to avoid overwriting your existing file.

Use bundles when you want to post a profile on the forum, send a setup to another pilot, or keep a portable backup that includes a little context about which simulator it was built for.

> Screenshot placeholder SC-04B: Profile header with export and import bundle icons.

LinkTool ships bundled **starter templates** for common simulators. Use them when you want curated example rules instead of an empty profile.

How to load a template:

1. On the Dashboard profile header, click **New profile from template** (template icon).
2. Pick a pack — for example **DCS starter**, **MSFS starter**, **LMU tyre temps**, **iRacing tyre temps**, or an **Elite Dangerous** pack (**status**, **combat**, **exploration**, **SRV**, or **Odyssey on foot**).
3. Click **Load template**.
4. Edit rules as needed, then **Save As** to store your copy.

Templates are read-only packs inside the application folder. Loading a template replaces the current in-memory profile; save under your own filename so you do not lose changes.

If **Settings → General → Autoload LED profiles** and **Load starter template when no profile exists** are both enabled, LinkTool can load the matching bundled template automatically when you select a simulator that has no remembered profile yet.

> Screenshot placeholder SC-04A: Profile template picker with simulator starter packs.

WHEN TO USE NEW PROFILE

`New profile` is most useful when:

- you want to start from scratch;
- you want a temporary test profile without old rules affecting the result;
- you do not want to risk overwriting a previously opened profile.

For a new untitled profile, it is usually best to use `Save As` as soon as the basic structure is ready.

AUTO-SAVE AND MANUAL SAVE

If profile auto-save is enabled in Settings, LinkTool will save changes automatically. If auto-save is disabled, save manually after meaningful changes.

AUTOLOAD BY SIMULATOR

If **Settings → General → Autoload LED profiles** is enabled, LinkTool remembers the last opened or saved profile **separately for each simulator**, not as one global last file.

Typical multi-simulator workflow:

1. Enable autoload in **Settings → General**.
2. Select **DCS World**, open your DCS `.led.json` profile, and save if needed.
3. Select **Microsoft Flight Simulator 2020**, open your MSFS profile.
4. Later, switch simulators on the Dashboard — LinkTool loads the matching profile automatically.

Open or save a profile while a simulator is selected to associate that file with that simulator. If a simulator has no remembered profile yet, the editor starts empty until you open or save a profile for that simulator — unless **Load starter template when no profile exists** is enabled in **Settings → General** (see below), in which case LinkTool loads the bundled starter template for that simulator.

A good working habit is:

1. Create or open a profile.
2. Configure one rule.
3. Test it.
4. Save.
5. Only then move on to the next complex rule.

That makes it much easier to identify when something changed in the wrong direction.

WHAT A PROFILE FILE ACTUALLY IS

For everyday use, think of a profile as a saved set of rules.

The `.led.json` file stores things such as:

- the list of rules;
- target devices referenced by those rules;
- conditions, colors, and LED modes;
- comments and other editor-visible rule information.

What this means in practice:

- if you save the file, your rule logic is preserved outside the running application;
- if you copy the file, you can keep backup versions of working setups;
- if you open another profile, the current working rule set in the editor changes to that file.

WHERE PROFILES ARE SAVED

LinkTool does not force one fixed user profile directory for normal manual work. The profile lives where you save the `.led.json` file.

That is why `Save As` matters:

- it lets you decide where the working file should live;
- it is the safest way to create named variants such as `DCS-test.led.json`, `MSFS-night.led.json`, or `WarThunder-debug.led.json`;
- it reduces the chance of overwriting a trusted working profile.

If you maintain several simulator-specific profiles, use clear filenames and keep them in a folder structure that mirrors how you actually use them.

> Screenshot placeholder SC-04: New profile, New profile from template, Open, Save, and Save As actions.

SECTION 12

A rule defines when and how LED behavior should change.

RULES AND THE RULE EDITOR

COMMON RULE TYPES

In day-to-day use, these are the most common categories:

- telemetry rule: react to a simulator parameter;
- **gradient rule**: map a numeric telemetry value to LED color using ordered color stops (for example tyre or brake temperature per wheel);
- button rule: react to a button press or release;
- axis rule: react to an axis value;
- keyboard rule: react to a keyboard shortcut.

WHAT A RULE USUALLY CONTAINS

Depending on the rule type, a rule normally includes:

- a target device;
- a parameter or event source;
- a trigger condition;
- one LED, several LEDs, or all LEDs;
- an LED mode;
- one or two colors;
- an optional comment.

Gradient rules replace a single threshold condition and Color A/B with one telemetry parameter and an ordered list of **color stops** (numeric value + color). LED mode is not used — the color is computed from the live parameter value.

THE SAFEST FIRST RULE

The most reliable first setup looks like this:

1. Create one simple telemetry rule.
2. Choose a parameter that clearly changes and is visible in the telemetry snapshot.
3. Use a simple condition such as equality or a threshold.
4. Target one specific LED.
5. Use a strong, obvious color.
6. Save the rule.
7. Check the result in the simulator.

A FAST WAY TO READ ANY RULE

If the rule editor feels dense at first, read each rule in this order:

1. When should it trigger.
2. Which device should react.
3. Which LED or LEDs should react.
4. What should the visible result be.

In practical terms, this means:

- start with the source parameter or event;
- then check the condition;
- then check the target device and LED selection;
- then confirm the color and mode;
- only after that look at comments, duplication, or fine-tuning.

This reading order is especially useful when you return to an older profile or need to understand a larger imported ruleset.

PRACTICAL RULE SETUP SCENARIOS

These three scenarios are the most useful starting points.

SCENARIO 1. A SIMPLE TELEMETRY RULE FOR A STATE INDICATOR

Use this when:

- you want one LED to react to a specific simulator state;
- the parameter is already visible in the telemetry snapshot.

Recommended flow:

1. Create a new telemetry rule.
2. Select the target device.
3. Select a parameter that already appears in the telemetry snapshot.
4. Set a simple condition such as `==`, `>=`, or `<=`.
5. Choose one LED for the first test.
6. Use a bright, obvious color.
7. Save the rule and test it in the simulator.

Expected result:

- the LED changes only when the parameter actually reaches the configured condition.

What usually goes wrong:

- the wrong parameter name is used;
- the condition compares against the wrong value format;
- the wrong LED or target device is selected.

SCENARIO 2. A BUTTON RULE FOR A CLEAR MANUAL EVENT

Use this when:

- you want to confirm that basic rule logic and device targeting work;
- you want lighting to react to a button press;
- telemetry is not configured yet, but you still want to confirm that rules and LEDs work.

Recommended flow:

1. Create a button rule.
2. Choose the button source device.
3. Enter the button manually or use detect if that is more convenient.
4. Choose **Button mode** (see **What Button Mode Means** below); for a first test, use **Pressed**.
5. Assign one LED and a bright color.
6. Save the rule.
7. Press the button and verify the response.

Expected result:

- the LED reacts to a clear, repeatable event without depending on simulator telemetry.

Why this scenario is useful:

- it separates telemetry issues from rule and device issues.

SCENARIO 3. AN AXIS RULE FOR THRESHOLD INDICATION

Use this when:

- you want an LED to react when an axis crosses a threshold;
- you need a simple warning-style indication.

Recommended flow:

1. Create an axis rule.
2. Choose the axis source device.
3. Select the axis.
4. Choose an operator such as `>=`.
5. Set the threshold.
6. Assign the LED and color.
7. Save the rule and move the axis slowly across the threshold.

Expected result:

- the LED changes in a predictable way when the threshold is crossed.

What to watch for:

- if the axis is noisy near the threshold, the LED may switch frequently;
- for the first test, choose a clearly separated threshold instead of a borderline value.

SCENARIO 4. A GRADIENT RULE FOR CONTINUOUS VALUES (TYRE OR BRAKE TEMPERATURE)

Use this when:

- one numeric parameter should drive LED color across several states (for example cold, optimal, overheating);
- separate threshold rules would need many entries for the same LED (for example four tyres × several temperature bands).

Recommended flow:

1. Create a new rule and set **Rule type** to **Gradient**.
2. Select the target device and **one LED** (one wheel corner per rule).
3. Choose the telemetry parameter (for LMU or iRacing tyres: `TyreTempFL`, `TyreTempFR`, `TyreTempRL`, or `TyreTempRR`; similar names exist for brake temperatures in many racing sims).
4. Review the default **color stops** (value + color). LinkTool seeds four tyre-temperature stops when you first switch to **Gradient**; tune the values while watching the telemetry drawer on track.
5. Use **Add stop** or **Remove** as needed (minimum two stops).
6. Optional: set **Gradient animation** to **Pulse**, **Blink**, or **Flash** and choose an **Animation alternate color** if you want the mapped color to animate between the gradient result and that second color.
7. Save the rule, then use **Duplicate rule** for each other corner — in the wizard, pick one LED and remap the parameter (for example `TyreTempFL` → `TyreTempFR`). See **Duplicate rule wizard** below.
8. Start LinkTool and verify on track.

Expected result:

- the LED color updates as the parameter moves between stops;
- values below the first stop or above the last stop clamp to the first or last stop color.

What to watch for:

- VIRPIL device LEDs use a **64-color palette**; gradients interpolate within that palette — smooth in practice, not unlimited RGB;
- gradient rules do not use **Rule mode** or separate Color A/B — color comes entirely from the stops;
- **Solid** keeps the mapped stop color static; **Pulse**, **Blink**, and **Flash** animate between the computed gradient color and **Animation alternate color**;
- if telemetry for the parameter is missing, the gradient rule does not apply and the LED follows your **LEDs without rules** fallback settings.

WORKING IN THE RULE EDITOR

The rule editor opens as a large overlay. This makes it easier to focus on the active rule instead of getting lost in the list.

Useful things to remember:

- if required condition fields are incomplete, saving the rule will be blocked;
- **Duplicate rule** opens a short wizard to clone a rule with new LED target(s) and optional parameter remap — see **Duplicate rule wizard** below;
- you can delete a rule;
- you can reorder rules, and the order matters for execution;
- you can target individual LEDs, clear the selection, or apply the rule to all LEDs;
- for **Gradient** rules, configure the parameter, color stops, and optional **Gradient animation** in the behavior panel; **Rule mode** and Color A/B are hidden because color is computed from stops (animation uses **Animation alternate color** when not **Solid**).

WHAT RULE MODE MEANS

For telemetry, button, and axis rules (not **Gradient** rules), **Rule mode** in the behavior panel controls how **Color A** and **Color B** are shown while the rule is active.

MODE	WHAT YOU SEE
Solid	Steady **Color A** while the rule matches
Blink	Alternates **Color A** and **Color B** at the global blink rate
Flash	Alternates **Color A** and **Color B** at the global flash rate
Pulse	Smooth fade between **Color B** and **Color A** at the default pulse rate
Breathe	Slower smooth fade between **Color B** and **Color A** — useful for subtle warnings that should not flash aggressively
Strobe	**Color A** for part of each cycle, then **Color B** — set **Strobe duty cycle (%)** on the rule
One-shot flash	A short flash sequence when the rule triggers, then off or alternate color

Gradient rules do not use **Rule mode**. They use **Gradient animation** (**Solid**, **Pulse**, **Blink**, or **Flash** on the mapped stop color) instead.

Global animation speeds are in **Settings → General** under **Blink And Flash Timing** and **Breathe frequency (Hz)**. If many rules feel too fast or too slow, adjust those values before rewriting individual rules.

Per-rule overrides:

- **Breathe** — optional **Breathe period override (Hz)**. Leave **0** to use the global **Breathe frequency (Hz)** from Settings.
- **Strobe** — **Strobe duty cycle (%)** controls how much of each cycle stays on **Color A** (1–99).

DUPLICATE RULE WIZARD

Use **Duplicate rule** when you need similar rules that differ mainly by LED target and, for telemetry rules, by parameter name — for example tyre-temperature gradients on each wheel corner.

Duplicate rule appears on each rule card and inside the expanded rule editor. It opens a modal wizard that keeps the source rule's logic unchanged (conditions, colors, gradient stops, button/axis triggers, latch behavior, and priority offset relative to the source) while you choose new targets.

Wizard steps:

1. **LED targets** — select one or more LEDs on the source rule's device. Each selected LED creates a separate duplicate rule inserted below the source. The source rule's current LED(s) are not pre-selected; pick at least one LED to continue.
2. **Parameter** (telemetry, gradient, and telemetry condition rules only) — keep the source parameter or remap it to another name from the active simulator list. Remap updates the rule **Parameter**, gradient parameter, matching condition rows, and latch-clear telemetry fields. **Button** and **Axis** rules skip this step.
3. **Review** — confirm a per-rule summary (LED + effective parameter). If a selected LED overlaps with an existing enabled rule at higher priority, LinkTool shows a non-blocking **Target conflicts** warning; you can still confirm.

On **Create**:

- LinkTool inserts one new rule per selected LED immediately below the source rule (order follows the LED chip order);
- the source rule editor collapses and the last inserted rule is selected and expanded;
- rule priorities re-sequence to match the visible list;
- **Cancel** closes the wizard with no changes.

Tips:

- for four tyre corners, run the wizard once per corner — pick one LED and remap the parameter each time (for example `TyreTempFL` → `TyreTempFR`);
- when several LEDs should share the same parameter and logic, select multiple LEDs in one wizard pass;
- if the simulator parameter list is unavailable, the wizard still lets you keep the source parameter; remap is disabled until telemetry names load.

> Screenshot placeholder SC-06B: Duplicate rule wizard on the Review step with LED summary and target conflict warning.

HOW RULE PRIORITY WORKS

Priority in LinkTool is defined by the rule order in the list.

That means:

- when you move a rule up or down, you are not only changing how it looks in the editor;
- you are changing its execution order;
- the visible list order is what decides priority;
- each rule's **Priority** number matches its position in the list (0 = top). You can edit the number directly, but drag-and-drop is the safer way to reorder.

Why this matters:

- if two rules can affect the same LED, their relative order can change the final visible result;
- if you keep a broad default rule and a very specific special-case rule, their order must be intentional;
- when you duplicate, insert, delete, or reorder rules, LinkTool re-sequences their priority to match the visible list.

Safe rule-ordering advice:

1. Put the most specific rules where you can find and understand them quickly.
2. Keep broad default rules separate from narrow warning rules.
3. After reordering, test the affected LED path again instead of assuming nothing changed.

Even if each individual rule looks correct, a bad order can still make the combined behavior look wrong.

A short overview of the same priority rules also appears under **Settings → General → Rule priority**.

TARGET CONFLICT BADGES

LinkTool performs a static check when your profile contains multiple **enabled** rules that target the same LED on the same device.

When a lower-priority rule would be shadowed by another rule on overlapping LED targets, LinkTool shows:

- a **target conflict** badge on the affected rule card;
- a conflict count chip in the Dashboard rules header when any conflicts exist;
- a tooltip that names the overlapping LEDs and the winning rule.

Click the badge on a conflicted rule to jump to the rule that currently wins that LED target.

Important details:

- conflict badges are advisory — they describe overlap in the profile, not live telemetry state;
- disabled rules are ignored by the conflict check;
- fixing a conflict usually means reordering rules, narrowing LED targets, or adjusting priorities so the intended rule runs first;
- runtime **Shadowed** badges during live evaluation use the same precedence rules (lower priority number wins).

> Screenshot placeholder SC-05A: Rule card with a target conflict badge and header conflict count chip.

WHAT BUTTON MODE MEANS

In the rule editor under **Trigger logic**, **Button mode** defines how LinkTool evaluates the selected button over time.

The three modes are:

- **Pressed** — the rule is active while the selected button is in the pressed state. Use this for momentary buttons and for toggle switches or multi-position controls that stay latched in one position (for example a throttle mode switch).
- **Released** — the rule is active while the selected button is not pressed. Use this when you want an LED indication for the idle or default position of a control.
- **Changed** — the rule is active only at the moment the button changes state (press or release). Use this for a brief flash or one-shot effect, not for a color that should stay latched while a switch remains in position.

LinkTool reads the current hardware button state from your VIRPIL device while LinkTool is running. For most setups, **Pressed** is the correct choice when the LED should stay lit according to the current switch or button position.

ONE LED, SEVERAL BUTTON RULES

You can assign multiple button rules to the same LED with different colors. LinkTool applies the first matching rule in the rules list (see rule order above). A typical pattern:

1. Rule A: LED blue when switch position 1 is **Pressed**.
2. Rule B: LED green when switch position 2 is **Pressed**.
3. Rule C: LED yellow when a momentary button is **Pressed** (or **Changed** if you only want a flash on press).

Put the most specific rule higher in the list if two conditions could overlap.

WHAT SHIFT MEANS IN BUTTON RULES

`Shift` is an extra condition for a button rule.

The simplest way to understand it is:

- without shift, one button event alone can trigger the rule;

- with shift enabled, the main button only counts when the chosen shift button is also active.

This is useful when you want one physical control to produce different lighting behavior depending on a second modifier input.

Typical use cases:

- one button acts normally in the default layer;
- the same button should trigger a different LED indication while a modifier button is held;
- you want the LED behavior to follow the same modifier logic you already use on the device or in the simulator.

Practical advice for first-time shift setup:

1. First make the basic button rule work without shift.
2. Verify the correct source device and button.
3. Only then enable shift and capture or enter the shift button.
4. Test the shifted combination deliberately, not by quick random button presses.

If a shifted rule never triggers, check these first:

- whether shift is enabled in that rule;
- whether the shift source device is the correct device;
- whether the shift button is correct;
- whether you are reproducing the exact modifier state expected by the rule.

WHILE EDITING

When the rule editor is open, LinkTool may pause normal rule execution. This is intentional and helps keep the editor state stable while you work, instead of letting live updates interfere with the form.

If LinkTool looks paused while you edit, that is expected.

COMMON RULE SETUP MISTAKES

- choosing a parameter that is not actually present in the current telemetry snapshot;
- choosing the wrong target device;
- sending the effect to the wrong LED;
- using a threshold or operator that does not match the incoming value format;
- expecting immediate results while LinkTool is not running yet or telemetry is not arriving yet.

PRACTICAL ADVICE

Do not build a large, complicated rule set in one pass. The safer pattern is:

1. create one rule;
2. test it;
3. save it;
4. use **Duplicate rule** or add the next one.

> Screenshot placeholder SC-05: Compact rules list with rule cards.

Screenshot placeholder SC-06: Open overlay rule editor. Screenshot placeholder SC-06B: Duplicate rule wizard (Review step).

SECTION 13

The dashboard includes a manual LED control area. It is one of the most useful tools for first-line validation.

MANUAL LED CONTROL

Use it when you want to:

- confirm that the device accepts lighting commands at all;
- test one specific LED quickly;
- verify a color without running a full rules setup;
- understand which physical LED corresponds to which target.

Typical flow:

1. Select the device.
2. Select the LED.
3. Choose a color.
4. Apply the color to the selected LED.
5. If needed, turn all LEDs on, turn all LEDs off, or reset LEDs.

If manual LED control works but rules do not, that is a strong sign that the USB path is working and the problem is more likely in the profile, telemetry source, or rule conditions.

> Screenshot placeholder SC-07: LED control bench.

SETTINGS AND INTEGRATIONS

SECTION 14

Think of Settings as the place where you define LinkTool's operating environment.

SETTINGS AND INTEGRATIONS

HOW TO OPEN AND USE SETTINGS CORRECTLY

Use the sidebar to open `Settings`. Once there, the page is organized into tabs and has a dedicated `Save settings` button in the header area.

That button matters. In practical use, the safe pattern is:

1. open the relevant tab;
2. make the change;
3. click `Save settings`;
4. go back to the Dashboard and validate the result.

Do not assume that changing a field and immediately returning to the Dashboard is enough for every workflow. Treat `Save settings` as an explicit step.

WHAT EACH SETTINGS TAB IS FOR

GENERAL

This tab controls application-wide behavior.

This is where you configure things such as:

- verbose logging;
- automatic profile loading **per simulator**;
- automatic profile saving;
- LED reset behavior on start and stop;
- fallback behavior for LEDs without rules in the active profile;
- update checking;
- automatic LinkTool startup;
- default LED color;
- local input polling interval;
- local UDP address and port;
- blink and flash timing;
- **Minimize to tray** — when enabled, minimizing LinkTool hides it from the taskbar and keeps it in the notification area; double-click the tray icon or choose **Show LinkTool** from the context menu to restore the window.

Use `General` when the question is "How should LinkTool behave overall?" rather than "How should one simulator connect?"

INTEGRATIONS

This tab is where simulator-specific connectivity is defined.

This is where you enter or manage:

- IP addresses;
- ports;
- timeouts;
- War Thunder poll interval and request timeout;
- file paths such as `status.json`;
- buttons that copy the needed support files for DCS and X-Plane;
- SimConnect polling behavior;
- shared memory poll interval and map name for Automobilista 2;
- shared memory poll interval for Assetto Corsa Evo, Le Mans Ultimate, and iRacing;
- UDP port and optional multicast listener for Project Motor Racing.

Use `Integrations` when the question is "Where does the data come from and where do the needed support files need to go?"

SHARED MEMORY

Open **Settings → Integrations** and select the **Shared Memory** tab in the transport matrix. This tab groups simulators that read telemetry from Windows shared memory instead of UDP, files, or SimConnect.

Inside **Shared Memory**, LinkTool shows one sub-tab per supported title:

- **Automobilista 2** — Project CARS 2 shared memory map (`$pcars2$`);
- **Assetto Corsa Evo** — built-in AC Evo physics, graphics, and static maps (`Local\acevo_pmf_*`);
- **Le Mans Ultimate** — official Studio 397 shared memory map (`LMU_Data`);
- **iRacing** — official IRSDK shared memory map (`Local\IRSDKMemMapFileName`).

Each panel includes:

- a short **setup checklist** for the game and LinkTool;
- **Poll interval (ms)** with a slider from `10` to `200` ms (default `50` ms);
- an **Advanced** section when map names or extra options need review;
- a footer note explaining that **no plugin install** is required for these simulators.

Typical workflow:

1. Configure the game first (for AMS2: **Options → System → Use Shared Memory = Project CARS 2**).
2. Open **Settings → Integrations → Shared Memory** and choose the simulator sub-tab.
3. Adjust the poll interval if needed, then click **Save settings**.
4. Select the same simulator on the Dashboard, start LinkTool, and verify telemetry in the drawer or **Telemetry lab**.

If shared-memory telemetry feels heavy on your PC during gameplay, raise the poll interval to **100 ms** or close the telemetry drawer / Telemetry lab while driving.

PROJECT MOTOR RACING

Open **Settings → Integrations** and select the **Project Motor Racing** tab. This simulator uses **binary UDP telemetry**, not shared memory.

The panel includes:

- a short **setup checklist** for PMR and LinkTool;

- **Port** (default `7576`, must match the game);
- optional **multicast listener** (`224.0.0.150`) when another app already owns the direct UDP port.

Typical workflow:

1. In PMR open **Settings** → **Preferences** and enable **UDP**, set **Host** to `127.0.0.1`, **Port** to `7576`, and **Frequency** to `60`.
2. Open **Settings** → **Integrations** → **Project Motor Racing** and confirm the port (enable multicast only if needed).
3. Click **Save settings**.
4. Select **Project Motor Racing** on the Dashboard, load a session on track, start LinkTool, and verify telemetry in the drawer or **Telemetry lab**.

> **Port conflict.** Only one application can bind UDP port `7576` in direct mode. Close Race Element or other PMR telemetry tools, or enable LinkTool's multicast listener and point the game at the multicast address instead.

AI INTEGRATIONS

This tab configures optional **remote AI** providers that explain telemetry parameters in plain language. LinkTool uses them for the **EXPLAIN** action in **Telemetry lab** and in the telemetry rule editor. EXPLAIN is read-only help — it does not write or change rules for you.

Open **Settings** → **AI Integrations** when you want help understanding an unfamiliar parameter name, typical units, or how a value might be used in an LED rule.

What you configure here:

- **Enable telemetry parameter explanations** — master switch for EXPLAIN;
- **Active provider** — which remote service LinkTool calls when you click EXPLAIN;
- **Groq**, **ChatGPT**, and **Claude** sub-tabs — paste an API key for each provider you might use (only the active provider needs a key at request time).

Typical setup:

1. Open **Settings** → **AI Integrations**.
2. Turn on **Enable telemetry parameter explanations**.
3. Choose **Active provider**. **Groq** is the default and the easiest free option.
4. Open the matching sub-tab and paste your API key.
5. Click **Save settings**.
6. On the Dashboard, select the simulator you are working with.
7. In **Telemetry lab** or the rule editor, click **EXPLAIN** (question/comment icon) next to a parameter.

API keys and billing

LinkTool talks to each provider through its **developer API**. That is a different product from chat apps and phone subscriptions.

PROVIDER TAB	SERVICE	WHERE TO CREATE AN API KEY	BILLING NOTES
Groq	Groq open models	console.groq.com	Free tier available; recommended default
ChatGPT	OpenAI API	platform.openai.com	Pay-as-you-go on your OpenAI Platform account
ChatGPT	Not chatgpt.com		ChatGPT Free, **Go**, Plus, and Pro do **not** include API keys or API credits
Claude	Anthropic API	console.anthropic.com	Pay-as-you-go on your Anthropic account
Claude	Not claude.ai chat only		Claude Pro subscription is separate from API billing

Practical guidance:

- For **no API billing**, keep **Groq** as the active provider and create a free Groq key.
- If you pay for **ChatGPT Go** on your phone, you still need a separate key from **platform.openai.com** to use the ChatGPT tab in LinkTool.
- Store API keys only in LinkTool settings on your PC. Treat them like passwords.

How EXPLAIN works

When you click **EXPLAIN**, LinkTool opens a dialog and sends a short request to the active provider. The request includes:

- your **selected UI language** (English, Russian, or French);
- the **currently selected simulator** title and category;
- the **parameter name** (and group name when available);
- the **current live value** when LinkTool already has it from telemetry.

The provider returns a concise explanation — usually two to four sentences — about what the value represents, typical units or range when known, and sometimes a practical LED-rule example.

LinkTool does **not** modify your profile from this dialog.

Response caching

LinkTool **caches explanations on disk** so repeat questions do not always call the API. If you ask about the same parameter again for the same simulator, language, and provider, LinkTool shows the stored answer immediately and marks it **Cached answer**.

Default cache file:

```
%LocalAppData%\VIRPIL Controls LinkTool\telemetry-explanation-cache.json
```

Each provider has separate cache entries. Switching from Groq to ChatGPT may fetch a fresh explanation for the same parameter.

Where EXPLAIN appears

LOCATION	HOW TO OPEN
Telemetry lab	Click **EXPLAIN** on a parameter row
Rule editor (telemetry condition)	Click **EXPLAIN** next to the parameter field

Recommended workflow:

1. Confirm telemetry is arriving for your simulator.
2. Find the parameter in **Telemetry lab**.
3. Click **EXPLAIN** if the name or value is unclear.
4. Create a rule or finish the condition once you understand the parameter.

> **Screenshot placeholder SC-08A:** Settings → AI Integrations with enable toggle, active provider, and API key field.
> **Screenshot placeholder SC-03B:** Telemetry explanation dialog with a cached answer.

SHIFT LINK

This tab replaces the legacy **VPC Shift Tool** for forwarding external shift, detent, zoom, and trim states between VIRPIL devices.

Use it when you need to:

- read shift-related state from one or two source devices;
- combine source channels with OR/AND rules;
- write the combined state to up to four destination devices;
- run Shift Link together with LinkTool LED output without a second utility fighting for the same HID device.

Typical workflow:

1. open **Settings → Shift Link**;
2. click **Import default VPC_SHIFT.ini** if you already used VPC Shift Tool, or **Browse INI file...** to pick a profile manually;
3. confirm source, destination, channel masks, and OR/AND rules;
4. enable **Enable Shift Link** and, if desired, **Start and stop with LinkTool**;
5. click **Save settings**;
6. start LinkTool runtime.

LinkTool stores Shift Link settings in `settings.json` under the `ShiftLink` section. Legacy import reads `%LOCALAPPDATA%\\VIRPIL\\VPC_SHIFT.INI` and maps saved VID/PID pairs to currently connected host devices.

Do not run VPC Shift Tool and LinkTool Shift Link against the same destination device at the same time.

UPDATES

This tab is for update-related workflow. Use it when you want to check build availability or complete an update process cleanly rather than troubleshooting rules.

DIAGNOSTICS

This tab is for logs and support preparation.

Use it when the question is "What exactly happened, and how do I collect evidence of it?"

WHAT YOU USUALLY CONFIGURE HERE

- general application behavior;
- profile auto-load and auto-save;
- startup behavior, LED reset on start/stop, and unruled-LED fallback preferences;
- integration settings for individual simulators;
- AI provider keys and the active provider for telemetry **EXPLAIN**;
- Shift Link routing between VIRPIL devices;
- logging and diagnostic options.

GENERAL SETTINGS: WHAT EACH IMPORTANT OPTION MEANS

The `General` tab is where many of the misunderstood options live. The following explanations are deliberately practical.

VERBOSE LOGGING

Use this when you need more detail in log output while diagnosing a problem.

Enable it when:

- a device is detected intermittently;
- telemetry seems inconsistent;
- copying support files seems to succeed but the app still behaves incorrectly;
- support asks you for richer logs.

Leave it off for ordinary use unless you actively need extra diagnostics.

AUTOLOAD LED PROFILES

Use this when you want LinkTool to reopen LED profiles automatically instead of loading them manually every time.

When enabled, LinkTool remembers the **last profile separately for each simulator**:

- on **startup**, it loads the profile associated with the currently selected simulator;
- when you **switch simulators** on the Dashboard, it loads that simulator's last profile (after any unsaved-change prompt).

Open or save a profile while a simulator is selected to associate that file with that simulator. If you maintain different `.led.json` files for DCS, MSFS, Elite Dangerous, and other simulators, enable this option and load each profile once per simulator — afterward, selecting the simulator is usually enough.

On upgrade from older builds that stored only one global last profile, LinkTool migrates that path to your last selected simulator automatically.

This is convenient for stable day-to-day setups across several simulators, but less ideal when you are experimenting and want a consciously clean start each session.

LOAD STARTER TEMPLATE WHEN NO PROFILE EXISTS

This option works together with **Autoload LED profiles**.

When both are enabled and LinkTool has **no saved profile yet for the active simulator**, it loads the bundled starter template for that simulator instead of starting with an empty rule list.

When disabled, a simulator without a remembered profile still opens with an empty editor until you create, open, or save a profile yourself.

Bundled packs include starters for **DCS World**, **Microsoft Flight Simulator 2020**, **Le Mans Ultimate**, **iRacing**, and five **Elite Dangerous** profiles (**status**, **combat**, **exploration**, **SRV**, **Odyssey on foot**). Use **New profile from template** to browse packs manually at any time.

AUTO-SAVE PROFILES

When enabled, LinkTool saves profile changes automatically after short edit bursts.

This is useful when:

- you make many small edits;
- you do not want to forget to save;
- the current profile is your active working document.

Leave it off when:

- you are testing risky rule changes;
- you prefer deliberate versioned saves;
- you want to compare alternative setups manually.

RESET LEDS BEFORE START

This controls whether LinkTool sends an explicit LED reset when you press **START** or **STOP**.

When this setting takes effect

- The toggle applies only at **START** and **STOP**.
- Changing it in **Settings** → **General** while LinkTool is already running does **not** change LEDs immediately.
- To compare ON vs OFF, change the setting, then press **STOP** and **START** again (or at least **START** after a state where LEDs differ).

When **enabled**:

- on **START**, LinkTool prepares connected LEDs before rule evaluation begins;
  - with **Firmware defaults** for unruled LEDs, this sends firmware reset command `100`, not a black PC override;
  - with **Configured color** or **Off (black override)** for unruled LEDs, this turns all LEDs off first;
- on **STOP**, LinkTool restores firmware default LED colors (command `100`).

When **disabled**:

- LinkTool leaves the current LED state unchanged on **START**;
- LinkTool does not restore firmware defaults on **STOP**.

Why ON and OFF can look the same

With **LEDs without rules = Firmware defaults**, both positions often look identical on **START** if the panel already shows firmware colors. In that case:

- **Off** leaves the current firmware look unchanged;
- **On** sends command `100`, which restores the same firmware colors again.

That is expected, not a broken setting.

When the difference is visible

- **STOP while a rule is still active** — with reset **On**, LEDs return to firmware on stop; with reset **Off**, they can stay on the last rule color.
- **After manual LED testing** — reset **On** at **START** clears the test color; reset **Off** keeps it until a rule or another action changes it.
- **Configured color** or **Off (black override)** for unruled LEDs — reset **On** at **START** visibly turns all LEDs off or repaints the background before rules run.

This option is independent from the **LEDs without rules** fallback mode below. Use it when you want an explicit reset at runtime start/stop, not when you want to control LEDs that have no rule in the profile.

While LinkTool is running, ruled LEDs in **Firmware defaults** mode return to firmware when their condition stops matching; that runtime behavior is separate from this start/stop toggle.

RESTORE LEDS ON EXIT

When enabled, LinkTool applies your **LEDs without rules** fallback mode to connected devices when the application closes.

This helps leave hardware in a predictable state after you quit LinkTool — for example firmware defaults, a configured background color, or an explicit off override — depending on the fallback mode you selected in **Settings → General**.

This is separate from **Reset LEDs Before Start**, which only runs when you press **START** or **STOP** while LinkTool is running.

Changing this setting does not alter LEDs immediately; it takes effect the next time you close the application.

DISABLE USB OUTPUT (DRY-RUN)

When enabled, LinkTool still evaluates rules and updates on-screen previews, but **does not send LED frames to connected hardware**.

Use this when:

- you want to test rule logic and previews without changing physical LEDs;
- you are debugging telemetry or profile behavior on a live device without disturbing your current panel lighting;
- support asks you to verify rule output in software only.

What still works in dry-run:

- rule compilation and live evaluation while LinkTool is running;
- device preview and other UI feedback driven by the resolved LED frame;
- manual LED tests from the dashboard (these still use USB unless you avoid them).

What changes:

- ruled LED output is not written to the device over USB;
- the Dashboard rules header shows a **Dry-run** chip while the option is enabled.

Disable dry-run before you expect hardware LEDs to follow your profile again.

LEDS WITHOUT RULES

This controls what happens to LEDs that have **no rule** in the active profile while LinkTool is running.

Available modes:

- **Firmware defaults** — LinkTool does not override those LEDs. They keep the colors and behavior defined by device firmware.

- **Configured color** — LinkTool paints all unruled LEDs with the selected **Default color** value below.
- **Off (black override)** — LinkTool explicitly forces unruled LEDs off through PC override.

Important details:

- this affects LEDs with **no rule** in the active profile;
- in **Firmware colors** mode, LEDs **with** rules also return to firmware defaults when no condition matches (LinkTool sends firmware reset command `100` for that device when PC override is released);
- in **Configured color** and **Off (black override)** modes, ruled LEDs still use the rule's secondary color (`Color Two`) when no condition matches;
- if **Configured color** is selected but the color value is invalid, LinkTool falls back to **Firmware defaults**.

Typical choice:

- use **Firmware defaults** when you want LinkTool to manage only the LEDs you explicitly wrote rules for;
- use **Configured color** when you want a uniform background on unused LEDs;
- use **Off (black override)** when you want unused LEDs explicitly dark while LinkTool is running.

DEFAULT COLOR

This color is used in two places:

1. as the paint value for **LEDs without rules** when fallback mode is **Configured color**;
2. as a convenient starting color when you create or edit rules in the rule editor.

It does not force every rule to use the same color, and it is ignored for unruled LEDs when fallback mode is **Firmware defaults** or **Off (black override)**.

CHECK FOR UPDATES

This controls whether LinkTool participates in normal update checking behavior.

It affects maintenance workflow, not profile logic.

AUTO-START LINKTOOL

This controls whether LinkTool starts working automatically after application startup or after the first connected VIRPIL device appears.

Enable it when you want a more appliance-like experience. Disable it when you want to inspect the current setup before rule execution begins.

POLL RATE, LOCAL UDP ADDRESS, AND LOCAL UDP PORT

These values define how local LinkTool services check for data and communicate.

Change them carefully and only when you understand the reason, for example:

- you need a different local address or port for a specific setup;
- you are troubleshooting a local UDP path;
- you want to tune how frequently local state is sampled.

BLINK, FLASH, AND BREATHE TIMING

These values set the default speed for LED animation modes across your rules:

- **Blink frequency (Hz)** — used by **Blink** rules and **Gradient animation → Blink**;

- **Flash frequency (Hz)** — used by **Flash** rules and **Gradient animation → Flash**;
- **Breathe frequency (Hz)** — used by **Breathe** rules unless a rule sets **Breathe period override (Hz)** to a non-zero value.

Pulse uses a built-in default rate. **Strobe** timing follows the blink frequency; only the on/off ratio is per-rule (**Strobe duty cycle (%)**).

If effects feel visually too slow or too aggressive, adjust these globals before rewriting large numbers of rules.

WHEN TO GO TO SETTINGS

Open Settings when:

- telemetry is missing;
- you need to define an address, port, file path, or another simulator-specific source;
- you want to change auto-load or auto-save behavior;
- you need more detailed logging;
- you switched simulators and want to verify the integration setup.

PRACTICAL INTEGRATION ORDER

1. Choose the first simulator you want to support.
2. Configure only that integration.
3. Confirm that telemetry is actually arriving.
4. Only then move on to rule creation.

That keeps integration problems separate from rule logic problems.

After changing Settings, explicitly save them before returning to the dashboard. Treat that as part of the normal integration flow, especially after editing addresses, ports, or polling intervals.

INTEGRATION QUICK MAP

Different simulators send data in different ways. That is normal: one may use UDP, another a status file, another gRPC, and another SimConnect.

The safest working model is:

- open Settings;
- configure only one active simulator first;
- verify the telemetry snapshot;
- only then build rules.

One practical consequence matters during troubleshooting: LinkTool only needs one working simulator path at a time. If you are debugging one integration, ignore the others until the active simulator is stable.

INTEGRATION SUMMARY TABLE

SIMULATOR	TRANSPORT	GAME-SIDE SETUP	LINKTOOL ACTION	DEFAULT VALUES
DCS World	UDP	Deploy hook script into `Saved Games\...\Scripts\Hooks`	IP, port, **Deploy DCS script**	`127.0.0.1:4123`
XPlane 11	UDP	FlyWithLua + LinkTool script in `Resources\plugins\FlyWithLua\Scripts`	IP, port, **Deploy FlyWithLua**, **Deploy X-Plane script**	`127.0.0.1:49000`
XPlane 12	Native UDP RREF	In X-Plane: Settings → Network → **Accept incoming connections**	Simulator IP/port, local UDP port, **X-Plane 12 Datarefs** page	`127.0.0.1:49000`, local `49001`
IL-2 Great Battles	UDP	`[KEY = telemetrydevice]` section in `data\startup.cfg`	IP and port in Settings	often `127.0.0.1:29373`
Elite Dangerous	`Status.json` + journal logs	game writes status and journal files	path to `Status.json`; optional **Journal folder**	`%UserProfile%\Saved Games\Frontier Developments\Elite Dangerous\`
War Thunder	HTTP	built-in local game server	IP, port, poll interval, request timeout	`127.0.0.1:8111`, poll `250` ms, timeout `500` ms
Automobilista 2	Shared memory	**Options → System → Use Shared Memory = Project CARS 2**	**Settings → Integrations → Shared Memory**	map `$pcars2$`, poll `50` ms
Assetto Corsa Evo	Shared memory	game running on track (built-in maps)	**Settings → Integrations → Shared Memory**	maps `Local\acevo_pmf_*`, poll `50` ms
Le Mans Ultimate	Shared memory	driving session on track (built-in `LMU_Data` map)	**Settings → Integrations → Shared Memory**	map `LMU_Data`, poll `50` ms

SIMULATOR	TRANSPORT	GAME-SIDE SETUP	LINKTOOL ACTION	DEFAULT VALUES
iRacing	Shared memory	live driving session on track (built-in IRSDK map)	**Settings → Integrations → Shared Memory**	map `Local\IRSDKMemMapFileName`, poll `50` ms
Project Motor Racing	UDP	**Settings → Preferences → UDP Enabled**, host `127.0.0.1`, port `7576`, frequency `60`	**Settings → Integrations → Project Motor Racing**	port `7576`, optional multicast `224.0.0.150`
MSFS / Prepar3D v5	SimConnect	simulator running	SimVars + poll interval	poll interval `250` ms

> **Legacy note. Flight Simulator X: Acceleration** is no longer supported in LinkTool v3. Saved settings that still reference FSX fall back to **Microsoft Flight Simulator 2020**.

> **Legacy note.** Older VPC Link Tool manuals also covered IL-2 Sturmovik 1946, Cliffs of Dover, and Desert Wings with separate `DeviceLink / conf.ini` sections. The current VIRPIL Controls LinkTool release supports **IL-2 Great Battles** out of the box. Use the legacy manuals for the other IL-2 family titles only if you are deliberately continuing an older Link Tool setup.

DCS WORLD

DCS World sends telemetry over UDP. The Lua hook script in `Saved Games` creates a UDP client and sends data to the address and port configured in LinkTool.

WHAT YOU CONFIGURE IN LINKTOOL

- IP address (`DCS World` in **Integrations**);
- UDP port;
- the **Deploy DCS script** button on the **Integrations** tab.

WHERE THE FILES GO

1. Open **Settings → Integrations**.
2. Set the IP address and port (defaults are `127.0.0.1` and `4123`).
3. Click **Deploy DCS script**.
4. When prompted, select the **DCS user folder root**, not the `Scripts` subfolder:
   - stable release: `%UserProfile%\Saved Games\DCS`;
   - open beta: `%UserProfile%\Saved Games\DCS.openbeta`;
   - if you moved your DCS profile elsewhere, point to that same folder.
5. LinkTool downloads the current script, copies it into `Saved Games\...\Scripts\Hooks`, and rewrites `config.lua` with the current IP and port.

Typical structure after deployment:

```
Saved Games\DCS.openbeta\
  Scripts\
    Hooks\
      vpcGameGUI.lua          ← entry point, runs when DCS starts
      vpc\
        config.lua            ← UDP host/port, VPC_DEBUG
        common.lua
        planes\               ← your plane-specific exports
        planes_examples\      ← ready examples for A-10C, FA-18C, etc.
          template.lua        ← boilerplate for new modules
```

WHAT THE MAIN FILES MEAN

- **Common aircraft telemetry** — parameters available for every official DCS module in single player and multiplayer.
- **Module cockpit telemetry** — cockpit arguments exported from files in `planes`. If a module file is missing, the script may create one from `planes_examples\template.lua` the first time you enter that cockpit.
- `config.lua` — the first manual check point. LinkTool writes:

```
VPC_DEBUG = false
VPC_UDP_SOCKET_HOST = '127.0.0.1';
VPC_UDP_SOCKET_PORT = 4123;
```

The host and port **must match** LinkTool Settings.

EXPANDING THE DCS PARAMETER SET

1. Open `Saved Games\...\Scripts\Hooks\vpc\planes_examples`.
2. Find an example for your module or copy `template.lua` into `planes\<module>.lua`.
3. Add the cockpit argument IDs you need to the returned Lua table.
4. Restart the mission or DCS.

For Lua debugging you can temporarily set `VPC_DEBUG = true`. That reloads user scripts when you leave and restart a single-player mission without restarting all of DCS. **Do not leave** `VPC_DEBUG = true` **enabled permanently:** it conflicts with TacView, SRS, and other hook utilities.

DCS VERIFICATION CHECKLIST

1. Set the IP address and port in LinkTool.
2. Click **Deploy DCS script** and choose the correct `Saved Games\DCS` or `DCS.openbeta` folder.
3. Click **Save settings**.
4. Start DCS World and load a mission or cockpit.
5. Select `DCS World` on the Dashboard, start LinkTool, and open the telemetry drawer.

IF TELEMETRY IS MISSING

Check in this order:

1. Do `vpcGameGUI.lua` and `vpc\config.lua` exist?
2. Do `VPC_UDP_SOCKET_HOST` and `VPC_UDP_SOCKET_PORT` match LinkTool Settings?

3. Is the port already in use by another application? If so, change the port in **both** LinkTool and `config.lua`.
4. Is the multiplayer server blocking aircraft data export?
5. Do you have the required plane-specific file in `planes` if you expect cockpit arguments rather than only common telemetry?

MANUAL INSTALLATION WITHOUT THE DEPLOY BUTTON

If automatic deployment is unavailable, download the script from the public repository vpc-linktool-dcs-script, extract it into `Saved Games\...\Scripts\Hooks`, and edit `vpc\config.lua` manually.

MICROSOFT FLIGHT SIMULATOR / MSFS 2024 / PREPAR3D V5

The SimConnect family does not use UDP scripts inside the game folder. LinkTool reads parameters through SimConnect, and the visible parameter set depends on your SimVar and LVar definitions.

WHAT YOU CONFIGURE IN LINKTOOL

- the active SimConnect-family simulator on the Dashboard;
- the SimConnect poll interval in **Settings → Integrations**;
- SimVar and LVar definitions on the **SimConnect Vars** page;
- optional aliases and normalization on the **Transforms** page.

SIMVARS AND LVARS

TYPE	SUPPORTED ON	HOW TO ADD
SimVar	MSFS, Prepar3D v5	**Add SimVar**, SDK variable name
LVar	MSFS 2020 (SU12+) and MSFS 2024	**Add LVar**, bare name without `L:`

Epsilon is an optional change threshold. A value of `0` accepts every update. Larger epsilon values reduce noisy telemetry on sensitive parameters such as airspeed or altitude. See the hint next to the Epsilon field in the variable editor for details.

WORKING ORDER

1. Select the correct SimConnect-family simulator on the Dashboard.
2. Open **SimConnect Vars** and enable the minimum set you actually need.
3. For third-party MSFS aircraft, add the required LVars with **Add LVar**.
4. Import or load variables if required:
   - **Load preset pack** — bundled **Generic cockpit**, **A320 cockpit**, or **Common LVars** starter set;
   - **Import** — LinkTool v3 SimVar backup;
   - **Import legacy Link Tool vars** — old `msfs2020.json`-style file, then click **Save SimVars**;
   - or add variables manually.
5. Return to **Settings → Integrations** and set the poll interval.
6. Click **Save settings**.
7. Start the simulator, then LinkTool, and inspect the telemetry drawer.

PRACTICAL ADVICE

- Do not enable dozens of new SimVars or LVars at once while you are still validating the basic stream.
- If values feel too noisy or too slow, adjust the **SimConnect poll interval** and **epsilon** before rewriting rules.

- If you need a cleaner alias or scaled value, create a transform on the **Transforms** page instead of duplicating logic across rules.

PREPAR3D V5

Supported scope: LinkTool currently provides **full** Prepar3D integration for **v5.x only**. Prepar3D v3, v4, and v6 are not supported in this release.

Prepar3D v5 uses the managed client `LockheedMartin.Prepar3D.SimConnect.dll` (x64). The legacy FSX `SimConnect.dll` is **not** valid for Prepar3D v5.

Compatibility rule: the SimConnect client must **not** be **newer** than the installed `Prepar3D.exe`. If a newer SDK line is merged into the simulator folder (for example SDK **5.4.9** with sim **5.3.17**), connection fails with `VERSION_MISMATCH`.

How LinkTool selects the DLL

1. Reads the `Prepar3D.exe` version from the registry / install path.
2. Collects candidates from:
   - `ThirdParty/SimConnect/prepar3d/versions/<release>/`
   - `Lockheed Martin/Prepar3D v5 SDK*` folders on local drives;
   - fallback `prepar3d/x64/` and the sim install path `lib/SimConnect/managed/`.
3. Picks the **newest compatible** DLL (client version ≤ simulator version).

The selected library is written to the system log, for example:
`Selected Prepar3D SimConnect library ... (Prepar3D v5 SDK 5.3.17.28160) for simulator 5.3.17.28160`.

Bundled clients shipped with LinkTool (Prepar3D v5)

SDK / SIM RELEASE	FOLDER INSIDE LINKTOOL
5.1.12.26829	`prepar3d/versions/5.1.12.26829/`
5.2.18.27565	`prepar3d/versions/5.2.18.27565/`
5.2.22.27615	`prepar3d/versions/5.2.22.27615/`
5.3.12.28072	`prepar3d/versions/5.3.12.28072/`
5.3.15.28130	`prepar3d/versions/5.3.15.28130/`
5.3.15.28131	`prepar3d/versions/5.3.15.28131/`
5.3.17.28160	`prepar3d/versions/5.3.17.28160/`
5.4.9.28482	`prepar3d/versions/5.4.9.28482/`

For sim **5.3.17.28160**, LinkTool automatically selects client **5.3.17.28160** (or the closest older compatible release if an exact match is unavailable).

SDK installation guidance

- Install the SDK in a **separate folder**, for example `J:\Lockheed Martin\Prepar3D v5 SDK 5.3.17.28160\`, **not** on top of `Prepar3D v5\`.

- The SDK line must **match** the simulator line or be **older** (client ≤ sim).
- If sim is **5.3.x** but only SDK **5.4.x** is available on the website, either upgrade sim to **5.4.x or** use an archived **5.3.x** SDK (LinkTool can also fall back to the bundled clients listed above).

Prepar3D and LVars

LVars are **not** polled on Prepar3D. They may remain in the saved **SimConnect Vars** profile, but Prepar3D runtime uses **SimVars only**.

IF PARAMETERS ARE MISSING

1. The simulator is actually running in a live flight or mission state.
2. The correct SimConnect-family simulator is selected.
3. The required SimVars or LVars are enabled and saved.
4. For LVars, the loaded aircraft actually publishes that variable.
5. LinkTool is started and active.

IL-2 GREAT BATTLES

IL-2 Great Battles sends telemetry over UDP. Unlike DCS and X-Plane, there is **no separate LinkTool script** to deploy. You configure the game first, then enter the same values in LinkTool.

GAME-SIDE SETUP

1. Find `startup.cfg` in the game installation folder, usually:

```
<IL-2 Great Battles install folder>\data\startup.cfg
```

2. Open it in a plain text editor.
3. Find or add this section:

```
[KEY = telemetrydevice]
      addr = "127.0.0.1"
      decimation = 2
      enable = true
      port = 29373
[END]
```

4. Save the file.
5. Fully restart IL-2 Great Battles if the game was already running.

The `addr` field is the IP address the game broadcasts to. It is usually `127.0.0.1`. The `port` field is the UDP broadcast port. It **must match** the port in LinkTool.

LINKTOOL SETUP

1. Open **Settings** → **Integrations**.
2. In the **IL-2 Great Battles** block, enter the same IP and port as in `startup.cfg`.
3. Click **Save settings**.
4. On the Dashboard, select `IL-2 Great Battles`, start LinkTool, and open the telemetry drawer.

COMMON IL-2 PROBLEMS

Empty telemetry drawer

- the port in `startup.cfg` does not match LinkTool Settings;
- the game was not restarted after editing `startup.cfg`;
- LinkTool is listening on a different address than the one in `addr`.

Noise, blank rows, or mixed values

A frequent cause is a UDP port collision with other software. The legacy manual often shows `telemetrydevice` and `motiondevice` sharing one port for SimShaker / JedPads:

```
[KEY = motiondevice]
    addr = "127.0.0.1"
    decimation = 2
    enable = true
    port = 29373
[END]
```

Fix:

- assign **different ports** to `telemetrydevice` and `motiondevice`;
- update the port in LinkTool;
- reconfigure the other software to its own port.

Parameters appear only in combat or a live mission

That is normal. Wait for an active gameplay state rather than the main menu.

XPLANE 11

XPlane 11 uses UDP through the FlyWithLua plugin and the LinkTool Lua script. This is a **two-step** deployment.

STEP 1. FLYWITHLUA

1. Open **Settings** → **Integrations**.
2. Set the X-Plane IP address and port (defaults are `127.0.0.1:49000`).
3. Click **Deploy FlyWithLua**.
4. Select the **X-Plane root folder** that contains `Resources\plugins`.
5. LinkTool downloads the framework into:

```
X-Plane\
  Resources\
    plugins\
      FlyWithLua\
```

STEP 2. LINKTOOL SCRIPT

1. On the same tab, click **Deploy X-Plane script**.
2. Select the X-Plane root folder again.

3. LinkTool deploys files into:

```
X-Plane\
  Resources\
    plugins\
      FlyWithLua\
        Scripts\
          script.lua
          config.lua.tpl
          variables.json
          variables.json.template
```

4. If `variables.json` is missing, LinkTool creates it from `variables.json.template`.
5. Click **Save settings**.
6. Restart X-Plane.

UDP SYNCHRONIZATION

`config.lua.tpl` contains the endpoint X-Plane sends data to:

```
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 49000
```

These values must match the X-Plane IP and port in LinkTool. The deploy button rewrites the file automatically.

EXPANDING THE X-PLANE PARAMETER SET

1. Open the X-Plane Datarefs reference.
2. Find the dataref you need, for example `sim/aircraft/view/acf_tailnum`.
3. Edit `variables.json`. It is a JSON array of objects with `name` and `ref` keys:

```
[
  {
    "name": "acf_tailnum",
    "ref": "sim/aircraft/view/acf_tailnum"
  }
]
```

4. Reload the Lua scripts from FlyWithLua inside X-Plane, or restart the simulator.
5. Confirm the new parameter in the LinkTool telemetry drawer.

X-PLANE VERIFICATION CHECKLIST

1. Deploy FlyWithLua.
2. Deploy the X-Plane script.
3. Save LinkTool settings.
4. Start X-Plane and load a flight.
5. Select `XPlane 11`, start LinkTool, and open the telemetry drawer.

IF TELEMETRY IS MISSING

- does `Resources\plugins\FlyWithLua` exist?
- do `script.lua`, `config.lua.tpl`, and `variables.json` exist in `FlyWithLua\Scripts`?
- do `SERVER_HOST` and `SERVER_PORT` match LinkTool Settings?
- was X-Plane restarted after deployment?

MANUAL INSTALLATION

If needed, install the components manually from xplane-flywithlua and vpc-linktool-xplane-script while preserving the same folder layout.

XPLANE 12

X-Plane 12 connects to LinkTool **without FlyWithLua and without scripts**. LinkTool asks the simulator for the parameters you need through the built-in UDP **RREF** protocol and reads the replies directly.

IN PLAIN TERMS: HOW THE CONNECTION WORKS

1. In X-Plane 12 you allow external UDP connections.
2. In LinkTool you specify **where** to connect (X-Plane IP and port) and **which parameters** to read (the dataref list).
3. When LinkTool is running and **XPlane 12** is selected on the Dashboard, the app subscribes to those datarefs and updates the telemetry drawer.
4. LED rules use the **telemetry names** from the dataref list, not the raw `sim/...` path.

Typical layout when the game and LinkTool run on the same PC:

```
LinkTool  --RREF subscribe-->  X-Plane 12 :49000
LinkTool  <--dataref values--  X-Plane 12
(local UDP :49001)
```

> **XPlane 11 or XPlane 12?** If you run **X-Plane 12**, choose **XPlane 12** on the Dashboard and **do not** deploy FlyWithLua — that is the older X-Plane 11 path. If you run **X-Plane 11**, use the **XPlane 11** section above.

STEP 1. CONFIGURE X-PLANE 12 (ONE-TIME)

1. Start X-Plane 12.
2. Open **Settings** → **Network**.
3. Enable **Accept incoming connections**.
4. Save and **load a flight** — do not stay in the main menu without an active flight.

If the game and LinkTool are on **different PCs** in the same LAN, X-Plane must run on the target machine and LinkTool must use that machine's IP instead of `127.0.0.1`.

STEP 2. TRANSPORT SETTINGS IN LINKTOOL

1. Open **Settings** → **Integrations**.
2. Find the **X-Plane 12** block.
3. Fill in the fields (defaults are usually fine on one PC):

FIELD	MEANING	DEFAULT
IP address	Where LinkTool sends RREF requests	`127.0.0.1`
Port	X-Plane **legacy** UDP port for native RREF	`49000`
Local UDP port	Windows port where LinkTool waits for replies	`49001`
RREF frequency (Hz)	How often X-Plane sends updates	`5`

4. If X-Plane is already running, click **Discover X-Plane on LAN** to auto-fill the address.
5. Click **Save settings** at the bottom of Settings.

> **Important:** LinkTool **does not hardcode** these values at runtime. It uses exactly what you enter under **Settings → Integrations → X-Plane 12** and save with **Save settings**. After you change IP, port, or local port, save and restart LinkTool (STOP → START).

> **Important:** the local UDP port (`49001`) is **not** the X-Plane port. You do not enter it inside the simulator. If the port is already in use, pick a free one (for example `49002` or `49003`) and save settings again.

LEGACY PORT VS X-PLANE NETWORK SCREEN

In **X-Plane 12 → Settings → Network** you may see several UDP values. **Do not copy them blindly into LinkTool.**

X-PLANE NETWORK LABEL	TYPICAL VALUE	SET IN LINKTOOL INTEGRATIONS?
Port we receive on	`49010`	**No** — newer DATA output port; native **RREF does not use this**
Port we receive on (legacy)	`49000`	**Yes** — set **Port** to this legacy value
Port we send from (legacy)	`49001`	**No** — X-Plane sends from this port; pick a **different Local UDP port** (for example `49003`)

For native RREF, **Port** in LinkTool must be the **legacy receive port** (`49000` unless you changed it in X-Plane). **Do not** enter X-Plane's **Port we receive on** (`49010`) into LinkTool **Port**.

Discover X-Plane on LAN fills the simulator IP and the legacy RREF port from the network beacon. You can still change **Local UDP port** and **RREF frequency** explicitly before saving.

STEP 3. DATAREF LIST (WHAT TO READ FROM THE SIMULATOR)

1. In the LinkTool sidebar, open **X-Plane 12 Datarefs**.
2. Use the **search field** to filter long dataref lists by telemetry name, dataref path, or live value.
3. Review the starter set or add your own entries:
   - **Telemetry name** — short name used in rules (for example `airspeed_kts`);

- **Dataref path** — X-Plane path as used by native RREF (for example `sim/cockpit2/gauges/indicators/airspeed_kts_pilot`, or a custom plugin path such as `myplugin/cockpit/gear`). Or click **Load preset pack** to merge a bundled starter set (see below).

4. If you already have a `variables.json` from X-Plane 11 / FlyWithLua, click **Import variables.json**.
5. Click **Save datarefs** on that page.

LinkTool validates enabled entries before **Save datarefs** and while you edit the list:

- **Telemetry name** is required.
- **Dataref path** is required.
- The path must not contain spaces or other whitespace/control characters.
- Two enabled entries cannot share the same telemetry name.

LinkTool does **not** require paths to start with `sim/`. Built-in simulator paths usually use `sim/cockpit2/...`, but aircraft add-ons and plugins may publish custom paths without that prefix. LinkTool sends the path to X-Plane RREF exactly as entered.

If X-Plane does not recognize a path, LinkTool still saves it, but the telemetry drawer may stay empty for that parameter until you fix the path for the loaded aircraft or plugin.

Reference: X-Plane Datarefs. Built-in simulator paths in X-Plane 12 usually use `sim/cockpit2/...` instead of older `sim/cockpit/...` paths. Custom aircraft and plugin paths may omit the `sim/` prefix.

Example `variables.json` entry (same meaning as in the LinkTool editor):

```
[
  {
    "name": "airspeed_kts",
    "ref": "sim/cockpit2/gauges/indicators/airspeed_kts_pilot"
  }
]
```

LOADING BUNDLED PRESET PACKS

LinkTool ships curated starter dataref sets for X-Plane 12.

1. Open **X-Plane 12 Datarefs**.
2. Click **Load preset pack**.
3. In the picker, choose a pack — for example **GA cockpit** (generic C172-style instruments) or **Zibo 737** (community Zibo-mod starter paths).
4. Click **Load pack**. LinkTool merges new rows into your list. Entries that already match the same telemetry name or dataref path are skipped.
5. Review the status message for added and skipped counts, then click **Save datarefs**.

Preset packs are curated starters. Aircraft add-ons may use different dataref paths; verify paths for your loaded aircraft before building rules.

STEP 4. START LINKTOOL AND VERIFY

1. Confirm X-Plane 12 is in an active flight.
2. On the **Dashboard**, select **XPlane 12** (not XPlane 11).
3. Press **START** / start LinkTool runtime.

4. Open the **telemetry drawer** — parameters from your dataref list should appear and change in flight.

X-PLANE 12 VERIFICATION CHECKLIST

1. X-Plane: **Accept incoming connections** is enabled.
2. LinkTool Integrations: IP `127.0.0.1`, **legacy RREF port** `49000` (not `49010`), local UDP `49001` or another free port — **Save settings**.
3. Datarefs page: at least one **enabled** dataref — **Save datarefs**.
4. Dashboard: **XPlane 12** selected, LinkTool **running**.
5. In flight, the telemetry drawer is **not empty**.

IF TELEMETRY IS MISSING

Check **in this order**:

1. **XPlane 12** is selected on the Dashboard, not XPlane 11.
2. LinkTool runtime is **running** (START).
3. X-Plane has **Accept incoming connections** enabled and a **flight** is loaded.
4. Integrations **Port** is the **legacy RREF port** (`127.0.0.1:49000` on one PC) — **not** X-Plane's **Port we receive on** (`49010`).
5. The **local UDP port** is free and not the same as X-Plane **Port we send from (legacy)** (`49001`); try `49003` if needed.
6. The **X-Plane 12 Datarefs** page has **enabled** entries and you clicked **Save datarefs**.
7. Dataref paths are recognized by the loaded aircraft or plugin (built-in XP12 paths often use `sim/cockpit2/...`; custom paths do not need a `sim/` prefix).
8. After transport or dataref changes, **restart LinkTool** (STOP → START).

If everything looks correct but values still do not appear, open **Settings → Diagnostics**, enable verbose logging, reproduce the issue, and attach the log through **Report issue**.

MIGRATING FROM X-PLANE 11

1. Export `variables.json` from `FlyWithLua\Scripts` or your old setup.
2. In LinkTool: **X-Plane 12 Datarefs → Import variables.json**.
3. Update dataref paths for XP12 if needed, then save the list.
4. You usually **do not** need to rename rule parameters in `.led.json` if the JSON `name` values match your rules.

ELITE DANGEROUS

Elite Dangerous does not use UDP. LinkTool reads `Status.json`, which the game updates periodically, and optionally tails **journal** log files in the same Saved Games folder for event-based `Journal.*` telemetry keys.

WHERE THE FILE LIVES

Typical path:

```
%UserProfile%\Saved Games\Frontier Developments\Elite Dangerous\Status.json
```

If the game was installed through another launcher or your Windows profile was moved, the path may differ.

LINKTOOL SETUP

1. Open **Settings → Integrations**.
2. In the **Elite Dangerous** block, use the file picker and select `Status.json`.

3. If available, use the open-folder shortcut to confirm you selected the correct directory.
4. Click **Save settings**.
5. Start Elite Dangerous, select the simulator on the Dashboard, and inspect the telemetry drawer.

JOURNAL FOLDER (OPTIONAL)

For combat and loadout events beyond `Status.json`, LinkTool can read Elite Dangerous **journal** `.log` files.

1. Open **Settings** → **Integrations**.
2. In the **Elite Dangerous** block, set **Journal folder** if auto-detection fails, or leave it empty to use the default Saved Games Elite Dangerous directory.
3. Click **Save settings**.

Journal keys appear in the telemetry drawer alongside `Status.json` values. When **Elite Dangerous** is active, the drawer groups parameters into sections such as **Journal**, **Ship**, **FSD**, **Fuel**, **SRV**, and **OnFoot**. They update when new journal lines are written (for example after damage, docking, or loading a save).

KEY	MEANING
`Journal.LastEvent`	Name of the most recent parsed journal event
`Journal.LastEventTime`	ISO timestamp from the most recent parsed journal line
`Journal.UnderAttack`	`1` while under attack; cleared on FSD jump, dock, or undock
`Journal.HeatDamage`	`1` on heat damage event; cleared like **UnderAttack**
`Journal.HeatWarning`	`1` on **HeatWarning**; cleared on FSD jump, dock, or undock
`Journal.HullHealth`	Hull health percent from **Damage** / **HullDamage** events
`Journal.ShieldsHealth`	Shield health percent from **Damage** events
`Journal.ShieldsUp`	`1` when shields are up, `0` when down (**ShieldState**)
`Journal.Interdicted`	`1` after **Interdicted**; cleared on escape, FSD jump, dock, or undock
`Journal.InterdictedSubmitted`	`1` if you submitted to the interdiction, `0` if you fought
`Journal.CockpitBreached`	`1` on **CockpitBreached**; cleared like **UnderAttack**
`Journal.ShipDestroyed`	`1` on **Died**; cleared on **LoadGame**
`Journal.FuelScooping`	`1` while **FuelScoop** events arrive; cleared on FSD jump, dock, or supercruise exit
`Journal.FuelTotal`	Main tank tons after the latest **FuelScoop** event
`Journal.InSrv`	`1` after **LaunchSRV**, `0` after **DockSRV**
`Journal.InSupercruise`	`1` after **SupercruiseEntry**, `0` after **SupercruiseExit** or hyperspace **FSDJump**
`Journal.FsdJumpType`	Jump type from **StartJump** (`Hyperspace` , `Supercruise` , …)
`Journal.Ship`	Ship type from **LoadGame** / **Loadout** events
`Journal.StarSystem`	Star system from **FSDJump** / **Location**
`Journal.Body`	Body name from **FSDJump** / **Location**
`Journal.TargetLocked`	`1` when **ShipTargeted** reports `TargetLocked: true`
`Journal.Wanted`	`1` after **CommitCrime**; cleared on **LoadGame**
`Journal.Fined`	`1` on **Fined**; cleared on FSD jump or dock
`Journal.JetConeBoost`	`1` on **JetConeBoost**; cleared on FSD jump or dock
`Journal.UssDrop`	`1` on **USSDrop**; cleared on FSD jump or dock

Use these for rules that should react to journal events that `Status.json` does not expose cleanly on its own.

IF DATA IS MISSING

- the path points to the wrong `Status.json`;
- the journal folder path is wrong and `Journal.*` keys never appear (status keys may still work from `Status.json`);
- the game has not created the file yet (launch it at least once from the main menu);
- `Elite Dangerous` is not selected as the active simulator;
- LinkTool settings were not saved after changing the path.

WAR THUNDER

War Thunder exposes data through a **local HTTP server** built into the game. This is not a UDP integration.

WHAT YOU CONFIGURE

- IP address — usually `127.0.0.1`;
- port — default `8111`;
- **Poll interval (ms)** — how often LinkTool requests `/state` and `/indicators` (default `250`, range `50` – `2000`);
- **Request timeout (ms)** — how long each HTTP request may wait for a response (default `500`, range `100` – `5000`).

VERIFICATION ORDER

1. Set the address, port, poll interval, and request timeout in **Settings** → **Integrations** → **War Thunder**.
2. Click **Save settings**.
3. Start War Thunder and load a battle or training session.
4. Select `War Thunder`, start LinkTool, and open the telemetry drawer.

IF DATA ARRIVES INCONSISTENTLY

- increase the **request timeout** if values arrive in bursts or HTTP calls time out;
- lower the **poll interval** if updates feel delayed, or raise it if LinkTool feels too aggressive on your PC;
- confirm the game is in an active session, not only hangar/menu;
- keep the default port `8111` unless the game explicitly uses another one.

AUTOMOBILISTA 2

Automobilista 2 (AMS2) exposes telemetry through the same **Project CARS 2 shared memory** layout used by Project CARS 2 and related titles. LinkTool reads the Windows memory map on your PC. **No plugin or mod install is required.**

LinkTool uses **shared memory only** for AMS2. The game also exposes a UDP telemetry stream, but LinkTool does not consume UDP for this integration.

WHAT YOU CONFIGURE IN AMS2

1. Launch **Automobilista 2**.
2. Open **Options** → **System**.
3. Set **Use Shared Memory** to **Project CARS 2**.
4. Set **UDP Frequency** to **1** (recommended when running external telemetry tools alongside the game).
5. Set **UDP Protocol Version** to **Project CARS 2**.
6. Click **Save**.

> **Screenshot reference:** In the System menu, **Use Shared Memory** and **UDP Protocol Version** should both read **Project CARS 2**, with **UDP Frequency** at **1**.

Shared memory is written only while you are **on track** in an active driving session (practice, qualifying, race, test day, etc.). Menus, garage, and replay screens usually do not publish live data.

WHAT YOU CONFIGURE IN LINKTOOL

1. Select **Automobilista 2** in the top simulator bar.
2. Open **Settings → Integrations → Shared Memory → Automobilista 2**.
3. Confirm:
   - **Poll interval** — default **50 ms** (10–200 ms range);
   - **Shared memory map name** — default `$pcars2$` (leave unchanged unless you know the game build uses a different map).
4. Click **Save settings**.
5. Load AMS2 and drive on track.
6. Start LinkTool and open the telemetry drawer to confirm parameters update.

USEFUL TELEMETRY PARAMETERS FOR LED RULES

These names appear in the rule editor when **Automobilista 2** is selected:

PARAMETER	TYPICAL USE
`PitLimiterActive`	Pit limiter engaged
`YellowFlag` / `BlueFlag`	Flag states
`Rpm`	Engine RPM threshold or rev lights
`Speed`	Speed band indicators
`Gear`	Gear position
`FuelLevel`	Low fuel warning
`AbsActive` / `TcActive`	Driver aid indicators
`CarName` / `TrackLocation`	Session context in status text

Use **Transforms** if you need scaled or aliased values (for example, RPM as a 0–100 percentage).

SAMPLE PROFILE

A starter racing profile is included:

`data/Profile[Automobilista 2] Racing[].led.json`

Open it from the dashboard or copy it as a template for pit limiter, flag, and RPM rules.

VERIFICATION ORDER

1. AMS2 **System** settings saved with **Use Shared Memory = Project CARS 2**.
2. You are **on track**, not in the main menu.
3. **Automobilista 2** is selected in LinkTool.
4. **Settings → Integrations → Shared Memory** values saved.
5. LinkTool is **started** and the telemetry drawer shows changing values.
6. One simple rule (for example `PitLimiterActive == 1`) before building a full profile.

IF NO TELEMETRY APPEARS

Check these in order:

1. **Use Shared Memory** is **Project CARS 2**, not **OFF** or **Project CARS 1**.
2. You clicked **Save** in the AMS2 System menu after changing options.
3. AMS2 is running a **live session on track** (not paused in menus for long periods).
4. LinkTool has **Automobilista 2** selected and settings were saved on the **Shared Memory** tab.
5. Map name is still `$pcars2$` unless you deliberately changed it.
6. LinkTool and AMS2 run under the same Windows user session (shared memory is local to the PC).
7. Try a lower poll interval (for example **30 ms**) if values feel delayed — avoid going below **10 ms** without reason.

NOTES FOR OTHER RACING TITLES

The underlying reader targets the **Project CARS 2 shared memory layout**. Future LinkTool releases may expose the same transport for **Project CARS 2** and **Project CARS 3** when product support is enabled; AMS2 is the first supported title in this family.

ASSETTO CORSA EVO

Assetto Corsa Evo (AC Evo) publishes live telemetry through **three Windows shared-memory maps** built into the game. LinkTool reads all three segments and merges them into one parameter snapshot for LED rules and the telemetry drawer. **No plugin or mod install is required.**

MAP	PURPOSE
`Local\acevo_pmf_physics`	Speed, RPM, gear, pedals, pit limiter, ABS/TC, tyre/brake temps
`Local\acevo_pmf_graphics`	Lap times, fuel, flags, pit state, shift hints, HUD tyre data
`Local\acevo_pmf_static`	Track, session, and version metadata (written at session load)

These map names are **Evo-specific**. They differ from original Assetto Corsa, ACC, and AC Rally, which use the older `Local\acpmf_*` namespace.

Shared memory is active only while you are **on track** in a driving session. Menus, garage, and replay usually do not publish live data.

WHAT YOU CONFIGURE IN AC EVO

No in-game telemetry toggle is required for the built-in shared-memory API. Launch the game and load a session on track.

> **Game updates.** AC Evo shared-memory layout and map names may change between major game patches. If telemetry stops after a game update, check for a newer LinkTool release or report the issue with your game build version

(`AcEvoVersion` in the telemetry drawer).

WHAT YOU CONFIGURE IN LINKTOOL

1. Select **Assetto Corsa Evo** in the top simulator bar.
2. Open **Settings → Integrations → Shared Memory → Assetto Corsa Evo**.
3. Confirm **Poll interval** — default **50 ms** (10–200 ms range). The three map names are fixed; you do not need to edit them.
4. Click **Save settings**.
5. Load AC Evo and drive on track.
6. Start LinkTool and open the telemetry drawer to confirm parameters update.

Performance tip. AC Evo physics telemetry can update very frequently. If LinkTool or the telemetry drawer feels sluggish while driving, raise the poll interval to **100 ms** on the **Shared Memory** tab, or close the drawer during gameplay.

USEFUL TELEMETRY PARAMETERS FOR LED RULES

These names appear in the rule editor when **Assetto Corsa Evo** is selected:

PARAMETER	TYPICAL USE
`Rpm` / `RpmPercent` / `MaxRpm`	Rev lights, shift indicators
`IsChangeUpRpm` / `IsChangeDownRpm`	Shift hints from the game HUD
`Speed` / `DisplaySpeedKmh`	Speed band indicators
`DisplayGearLabel`	Human-readable gear (`N` , `R` , `1` , `2` , …)
`Gear`	Raw gear index (`0` = reverse, `1` = neutral, `2+` = forward)
`PitLimiterActive`	Pit limiter engaged
`AbsActive` / `TcActive`	Driver aid indicators
`FuelLiterCurrentQuantity` / `FuelPercent`	Low fuel warning
`Flag` / `GlobalFlag`	Session flag state
`IsInPitLane` / `IsInPitBox`	Pit lane / garage indicators
`GraphicsTyreTemperatureLf` … `Rr`	Tyre temps (match in-game HUD)
`Track` / `CarModel`	Session context

Tyre temperature note. AC Evo does not populate the legacy physics-block inner/middle/outer tyre fields. Use the `GraphicsTyreTemperature*` parameters for tyre-based rules.

> **Delta time note.** When no reference lap is available, `DeltaTimeMs` reports `0` instead of a raw sentinel value from the game.

Use **Transforms** if you need scaled or aliased values.

VERIFICATION ORDER

1. AC Evo is running a **live session on track**.
2. **Assetto Corsa Evo** is selected in LinkTool.
3. **Settings → Integrations → Shared Memory** poll interval saved.
4. LinkTool is **started** and the telemetry drawer shows changing values (`Rpm` , `Speed` , `FuelLiterCurrentQuantity` , etc.).
5. One simple rule (for example `PitLimiterActive == 1` or `Rpm > 7000`) before building a full profile.

IF NO TELEMETRY APPEARS

Check these in order:

1. AC Evo is on **track**, not in the main menu or replay.
2. **Assetto Corsa Evo** is selected in LinkTool and settings were saved.
3. LinkTool was **started after** you loaded the session (restart LinkTool if you changed game or LinkTool settings mid-session).
4. LinkTool and AC Evo run under the same Windows user session (shared memory is local to the PC).
5. After a **game patch**, confirm LinkTool supports your AC Evo build — check `AcEvoVersion` and `SmVersion` in the telemetry drawer when data is flowing.

LE MANS ULTIMATE

Le Mans Ultimate (LMU) publishes live telemetry through the official `LMU_Data` Windows shared-memory map (Studio 397 **SharedMemoryInterface**). LinkTool reads the player vehicle telemetry and scoring blocks and merges them into one parameter snapshot for LED rules and the telemetry drawer. **No plugin or mod install is required.**

Shared memory is active only while you are **on track** in a driving session. Menus, garage, and replay usually do not publish live data.

WHAT YOU CONFIGURE IN LMU

No separate in-game telemetry toggle is required for the built-in shared-memory API. Launch the game and load a session on track.

> **Game updates.** LMU shared-memory layout may change between major game patches. If telemetry stops after an update, check for a newer LinkTool release or report the issue with your game build version.

WHAT YOU CONFIGURE IN LINKTOOL

1. Select **Le Mans Ultimate** in the top simulator bar.
2. Open **Settings → Integrations → Shared Memory → Le Mans Ultimate**.
3. Confirm **Poll interval** — default **50 ms** (10–200 ms range). The map name `LMU_Data` is fixed; you do not need to edit it.
4. Click **Save settings**.
5. Load LMU and drive on track.
6. Start LinkTool and open the telemetry drawer to confirm parameters update.

> **Performance tip.** If LinkTool or the telemetry drawer feels sluggish while driving, raise the poll interval to **100 ms** on the **Shared Memory** tab, or close the drawer during gameplay.

USEFUL TELEMETRY PARAMETERS FOR LED RULES

These names appear in the rule editor when **Le Mans Ultimate** is selected:

PARAMETER	TYPICAL USE
`Rpm` / `MaxRpm`	Rev lights, shift indicators
`Speed`	Speed band indicators
`Gear`	Gear position
`Throttle` / `Brake` / `Clutch` / `Steering`	Input-based indicators
`FuelLevel` / `FuelCapacity`	Low fuel warning
`InPits` / `PitMode` / `PitState`	Pit lane indicators
`AbsActive` / `TcActive`	Driver aid indicators
`SpeedLimiterActive`	Pit limiter / limiter state
`TyreTempFL` … `TyreTempRR`	Tyre temperature — **Gradient** rules (one per corner/LED) or threshold rules for simple warnings
`BrakeTempFL` … `BrakeTempRR`	Brake temperature — same **Gradient** pattern as tyres
`TrackName` / `Session` / `RacePosition`	Session context
`YellowFlag` / `BlueFlag` / `GreenFlag`	Flag state

> **Tyre and brake temperature.** For multi-state colour per wheel, prefer **Gradient** rules on `TyreTempFL` … `TyreTempRR` or `BrakeTempFL` … `BrakeTempRR`. Four corners need **four rules**, not many separate threshold rules per state.

Use **Transforms** if you need scaled or aliased values.

VERIFICATION ORDER

1. LMU is running a **live session on track**.
2. **Le Mans Ultimate** is selected in LinkTool.
3. **Settings → Integrations → Shared Memory** poll interval saved.
4. LinkTool is **started** and the telemetry drawer shows changing values (`Rpm`, `Speed`, `FuelLevel`, etc.).
5. One simple rule (for example `InPits == 1` or `Rpm > 7000`) before building a full profile.

IF NO TELEMETRY APPEARS

Check these in order:

1. LMU is on **track**, not in the main menu or replay.
2. **Le Mans Ultimate** is selected in LinkTool and settings were saved.
3. LinkTool was **started after** you loaded the session (restart LinkTool if you changed game or LinkTool settings mid-session).
4. LinkTool and LMU run under the same Windows user session (shared memory is local to the PC).
5. After a **game patch**, confirm LinkTool supports your LMU build.

IRACING

iRacing publishes live telemetry through the official **IRSDK** Windows shared-memory map `Local\IRSDKMemMapFileName`. LinkTool reads the dynamic variable header table, selects the latest triple-buffered telemetry row, and normalizes values into LinkTool parameter names for LED rules and the telemetry drawer. **No plugin or mod install is required.**

Shared memory is active only while you are **on track** in a live driving session. Menus, garage, spectating, and replay usually do not publish live IRSDK data.

WHAT YOU CONFIGURE IN IRACING

No separate in-game telemetry toggle is required. Launch iRacing and load a session on track. IRSDK shared memory starts automatically when you are in the car on track.

WHAT YOU CONFIGURE IN LINKTOOL

1. Select **iRacing** in the top simulator bar.
2. Open **Settings** → **Integrations** → **Shared Memory** → **iRacing**.
3. Confirm **Poll interval** — default **50 ms** (10–200 ms range). The map name `Local\IRSDKMemMapFileName` is fixed; you do not need to edit it.
4. Click **Save settings**.
5. Load iRacing and drive on track.
6. Start LinkTool and open the telemetry drawer to confirm parameters update.

> **Performance tip.** If LinkTool or the telemetry drawer feels sluggish while driving, raise the poll interval to **100 ms** on the **Shared Memory** tab, or close the drawer during gameplay.

USEFUL TELEMETRY PARAMETERS FOR LED RULES

These names appear in the rule editor when **iRacing** is selected:

PARAMETER	TYPICAL USE
`Rpm`	Rev lights, shift indicators
`Speed`	Speed band indicators (IRSDK value is **m/s**, not km/h)
`Gear`	Gear position (−1 reverse, 0 neutral)
`Throttle` / `Brake` / `Clutch` / `Steering`	Input-based indicators
`FuelLevel` / `FuelLevelPct`	Low fuel warning
`OnPitRoad` / `PitSpeedLimiterActive`	Pit lane indicators
`dcABS` / `dcTractionControl`	Driver aid indicators
`TyreTempFL` … `TyreTempRR`	Tyre temperature — **Gradient** rules (one per corner/LED) or threshold rules
`LFtempCM` … `RRtempCR`	Additional per-corner tyre temps (middle/right tread)
`WaterTemp` / `OilTemp`	Engine temperature warnings
`Lap` / `LapCurrentLapTime` / `SessionState`	Session context
`YellowFlag` / `BlueFlag` / `GreenFlag` / `RedFlag` / `CheckeredFlag`	Flag state
`IsOnTrack` / `PlayerTrackSurface` / `PitsOpen`	On-track and pit context

> **Tyre temperature.** For multi-state colour per wheel, prefer **Gradient** rules on `TyreTempFL` … `TyreTempRR`. Four corners need **four rules**. LinkTool ships an **iRacing tyre temps** starter template with animated gradient examples — load it via **New profile from template**.

Use **Transforms** if you need scaled or aliased values (for example converting `Speed` from m/s to km/h).

VERIFICATION ORDER

1. iRacing is running a **live session on track** (in the car, not in menus).
2. **iRacing** is selected in LinkTool.
3. **Settings → Integrations → Shared Memory** poll interval saved.
4. LinkTool is **started** and the telemetry drawer shows changing values (`Rpm`, `Speed`, `FuelLevel`, etc.).
5. One simple rule (for example `OnPitRoad == 1` or `Rpm > 7000`) before building a full profile.

IF NO TELEMETRY APPEARS

Check these in order:

1. iRacing is **on track** in a live driving session, not in the main menu, spectating, or replay.
2. **iRacing** is selected in LinkTool and settings were saved.

3. LinkTool was **started after** you loaded the session (restart LinkTool if you changed game or LinkTool settings mid-session).
4. LinkTool and iRacing run under the same Windows user session (shared memory is local to the PC).
5. After an **iRacing or LinkTool update**, confirm LinkTool supports your build.

PROJECT MOTOR RACING

Project Motor Racing (PMR) sends **binary UDP telemetry** packets to the host and port configured in the game. LinkTool listens on that port, decodes race info, participant state, and vehicle telemetry for the player car, and exposes the values as rule parameters. **No plugin install is required**, but UDP must be enabled in the game.

PMR publishes three packet types on the same port: session/race info, participant race state, and per-vehicle telemetry. LinkTool merges them into one player snapshot for LED rules and the telemetry drawer.

WHAT YOU CONFIGURE IN PMR

1. Open **Settings** → **Preferences** in the game.
2. Set **UDP Enabled** to **On**.
3. Set **UDP Host** to `127.0.0.1` for local use on one PC.
4. Set **UDP Port** to `7576` (or another port you will also set in LinkTool).
5. Set **UDP Frequency** to `60` (recommended starting value).
6. Save preferences and load a driving session **on track**.

> **Documentation note.** PMR may ship UDP sample documentation in the game install folder. Use the in-game port and frequency values if they differ from the defaults above.

WHAT YOU CONFIGURE IN LINKTOOL

1. Select **Project Motor Racing** in the top simulator bar.
2. Open **Settings** → **Integrations** → **Project Motor Racing**.
3. Set **Port** to match the game (default `7576`).
4. Leave **multicast** disabled for typical single-PC use. Enable it only when another app already listens on the direct port and you configured PMR for multicast output (`224.0.0.150`).
5. Click **Save settings**.
6. Start LinkTool while on track and open the telemetry drawer to confirm parameters update.

> **Port conflict.** Only one application can bind the same UDP port in direct mode. Close Race Element, MoTeC, or other PMR telemetry tools before starting LinkTool, or use multicast mode on both sides.

USEFUL TELEMETRY PARAMETERS FOR LED RULES

These names appear in the rule editor when **Project Motor Racing** is selected:

PARAMETER	TYPICAL USE
`Rpm` / `MaxRpm` / `EngineRevRatio`	Rev lights, shift indicators
`Speed` / `SpeedMs` / `EstimatedSpeedKmh`	Speed band indicators
`Throttle` / `Brake` / `Clutch` / `Steering`	Input-based indicators
`Gear`	Gear position
`FuelLevel` / `FuelCapacity` / `FuelUseRate`	Fuel warnings
`InPits` / `SessionFinished`	Pit and session state
`AbsActive` / `TcActive` / `RevLimiterActive`	Driver aid indicators
`TyreTempFL` … `TyreTempRR`	Tyre temperature warnings
`TyrePressureFL` … `TyrePressureRR`	Tyre pressure indicators
`TrackName` / `Session` / `RacePosition`	Session context
`YellowFlag` / `Flags`	Flag state

Use **Transforms** if you need scaled or aliased values.

VERIFICATION ORDER

1. PMR **UDP Enabled** is on and host/port/frequency are saved in the game.
2. You are **on track** in a driving session.
3. **Project Motor Racing** is selected in LinkTool and **Settings → Integrations → Project Motor Racing** was saved.
4. No other app is using the same UDP port (unless multicast is configured on both sides).
5. LinkTool is **started** and the telemetry drawer shows changing values (`Throttle` , `Rpm` , `Speed` , etc.).
6. One simple rule (for example `Brake > 0.5` or `Rpm > 6500`) before building a full profile.

IF NO TELEMETRY APPEARS

Check these in order:

1. PMR **UDP Enabled** is **On** and **UDP Host** is `127.0.0.1` for local use.
2. The **port in PMR matches LinkTool** (default `7576`).
3. You are **on track**, not in menus.
4. **Project Motor Racing** is selected in LinkTool and settings were saved.
5. **Race Element** or another PMR tool is **not** already bound to the same port — close it or switch LinkTool to multicast.
6. LinkTool was **started after** you loaded the session (restart LinkTool if you changed UDP settings mid-session).
7. Windows Firewall is not blocking LinkTool's UDP listener on the chosen port.

PDF-FRIENDLY INTEGRATION CHECKLIST

For future PDF export, this short route is the most useful summary:

1. Select the simulator.
2. Configure only that simulator's connection settings.
3. Verify the telemetry snapshot.
4. Create one simple rule.
5. Verify the visible LED response.

END-TO-END EXAMPLE: FROM EMPTY PROFILE TO A WORKING LED

The following scenario works well both as a first-time tutorial and as a full end-to-end check.

Goal of the scenario:

- select one simulator;
- confirm that telemetry is arriving;
- create a clean profile;
- build one simple rule;
- see a real LED response on hardware.

Step 1. Prepare the data source:

1. Start LinkTool.
2. Click `Refresh devices` if the device list did not update on its own.
3. Confirm that the target device appears with the correct title.
4. Select the one simulator you want to work with right now.
5. Open Settings and configure only that simulator's integration.

Step 2. Verify telemetry:

1. Return to the Dashboard.
2. Open the telemetry drawer.
3. Confirm that parameters appear in the table.
4. Find one parameter that is easy to observe and intentionally change.

Step 3. Create a clean profile:

1. Click `New profile`.
2. Confirm that you are working in a fresh rule set.
3. Use `Save As` immediately if you want to keep this test profile separate.

Step 4. Configure one rule:

1. Add a new telemetry rule.
2. Select the target device.
3. Select the parameter you identified earlier.
4. Set a simple condition that is easy to reproduce.
5. Assign one specific LED.
6. Use a bright color and a simple mode such as solid.
7. Save the rule.

Step 5. Verify the result:

1. Start LinkTool if it is not active yet.
2. Change the simulator state so the rule condition becomes true.
3. Watch for the LED response.
4. If the LED does not react, check the telemetry snapshot first, then the device target, then the rule condition.

Expected outcome:

- you have a working profile;
- you validated the full chain from integration to visible LED behavior;
- you now have a safe baseline for more complex rules.

Why this scenario matters:

- it shows where the chain breaks if something does not work;
- once this works, adding more rules becomes much easier and safer.

> Screenshot placeholder SC-08: Settings page with tabs.

When something fails, the fastest path is usually to move from simple checks to more specific ones.

DIAGNOSTICS AND TROUBLESHOOTING

DIAGNOSTICS TAB: WHAT IT GIVES YOU

The Diagnostics tab gives you concrete troubleshooting tools, not just status text.

The most useful pieces are:

- the log directory path;
- the current log level;
- the number of log and archive files;
- the latest log file name;
- quick actions for opening the logs folder, opening the latest log, and exporting a logs archive.

This is the right place to go before filing a bug report, because it helps you collect the exact material support will usually ask for.

RECOMMENDED BUG REPORT WORKFLOW

When a problem is reproducible, use this order:

1. reproduce it once on purpose;
2. note the simulator, device, and profile involved;
3. open Diagnostics and collect logs if needed;
4. use the `Report issue` button in the top header;
5. describe what you expected and what happened instead.

If you can include the active profile name and whether manual LED control still works, your report becomes much easier to triage.

SCENARIO 1. THE DEVICE IS NOT VISIBLE

Check these in order:

1. The device is physically connected.
2. You clicked `Refresh devices`.
3. The device is visible in Windows.
4. The list refreshed after reconnecting the device.

If the device does not appear even after refresh, start with the connection layer rather than the rules.

SCENARIO 2. THE DEVICE IS VISIBLE BUT THE LEDS DO NOT RESPOND

Check:

1. Whether manual LED control works.

2. Whether the correct device is selected.
3. Whether the correct LED is selected.
4. Whether another profile or rule is currently overriding the expected effect.

SCENARIO 3. TELEMETRY IS NOT ARRIVING

Check:

1. Whether the correct simulator is selected.
2. Whether the simulator is actually running.
3. Whether the integration is configured in Settings.
4. Whether any values appear in the telemetry snapshot.

If the telemetry snapshot is empty, a rule that depends on simulator data will almost certainly fail as well.

SCENARIO 4. A RULE SAVES BUT NEVER TRIGGERS

Check:

1. Whether the rule is blocked by a condition validation issue.
2. Whether the parameter matches what is actually visible in the telemetry snapshot.
3. Whether the target device matches the real target.
4. Whether the threshold, range, or operator makes sense for the incoming value.
5. Whether LinkTool is started.

SCENARIO 5. AN IMPORTED LEGACY PROFILE LOOKS WRONG

After importing an older profile, review:

- device mapping;
- parameter names;
- target LEDs;
- condition logic in a few key rules.

Import support helps preserve older profiles, but a manual validation pass is still the safest choice.

SCENARIO 6. EXPLAIN DOES NOT WORK OR SHOWS AN ERROR

Check:

1. **Enable telemetry parameter explanations** is on under **Settings → AI Integrations**.
2. **Active provider** matches the sub-tab where you pasted an API key.
3. You clicked **Save settings** after entering the key.
4. For **ChatGPT** or **Claude**, your key is from the **developer platform** (OpenAI Platform / Anthropic Console), not from a chat subscription alone.
5. LinkTool has network access and the provider account has billing or free-tier quota available.
6. If the dialog shows **Cached answer**, LinkTool reused a prior response — that is normal.

If EXPLAIN is disabled or the key is missing, LinkTool shows a short message pointing you to **Settings → AI Integrations** rather than technical environment-variable names.

PRACTICAL TROUBLESHOOTING RECIPES BY SIMULATOR

These are short, practical checklists for the most common simulator-side failures.

DCS WORLD: THE DEVICE IS VISIBLE BUT TELEMETRY NEVER ARRIVES

Check these in order:

1. `DCS World` is selected as the active simulator.
2. The IP address and port are configured in Settings.
3. The DCS export script has been copied.
4. Any parameters appear in the telemetry snapshot.

If the snapshot is still empty after that, the problem is usually in the DCS setup rather than in the rule itself.

SIMCONNECT FAMILY: MSFS/PREPAR3D IS SELECTED BUT NO PARAMETERS APPEAR

Check these in order:

1. The correct SimConnect-family simulator is selected.
2. SimVars are configured and enabled.
3. The polling interval is reasonable.
4. The simulator is running in a live flight or mission state, not only the main menu.

If SimVars are not configured, LinkTool cannot produce a useful parameter stream even when the simulator selection is correct.

PREPAR3D V5: SIMCONNECT ERROR / VERSION_MISMATCH / EMPTY TELEMETRY

1. Compare the `Prepar3D.exe` version with the SimConnect client version. The client must **not** be newer than the sim (common case: sim **5.3.17**, SDK **5.4.9** merged into the sim folder).
2. Install the SDK **separately** from the sim directory (`Lockheed Martin/Prepar3D v5 SDK <release>/`).
3. Check the LinkTool system log for `Selected Prepar3D SimConnect library` with the DLL path and SDK release.
4. Confirm the sim is running **in flight**, not only on the main menu.
5. For sim **5.3.x**, LinkTool can use the bundled client from `prepar3d/versions/5.3.x/` without a separate SDK on disk.

If the log contains `client version ... is newer than the installed simulator`, the issue is a client/sim version mismatch, not the LED rules.

IL-2 GREAT BATTLES: THE GAME IS RUNNING BUT THE SNAPSHOT IS EMPTY

Check these in order:

1. Does the `[KEY = telemetrydevice]` section exist in `data\startup.cfg`?
2. Do `addr` and `port` in `startup.cfg` match LinkTool Settings?
3. Was the game restarted after editing `startup.cfg`?
4. Is `IL-2 Great Battles` the active simulator?
5. Does the snapshot change after a live mission or gameplay state begins?
6. Is another telemetry or motion tool using the same UDP port as `telemetrydevice`?

If the configuration is incomplete, LinkTool may sit and wait but still never receive usable telemetry.

If values appear as noise, partial rows, or obviously mixed data, suspect a connection conflict before rewriting the rules.

ELITE DANGEROUS: THE SIMULATOR IS SELECTED BUT NO DATA APPEARS

Check these in order:

1. The `status.json` path is correct.
2. The game or status-file location has not changed.
3. LinkTool can still resolve that folder and the snapshot starts receiving values.

If the path is wrong, the rule logic is not the problem. The status source must exist first.

XPLANE 11: SETTINGS ARE FILLED IN BUT TELEMETRY IS STILL SILENT

Check these in order:

1. The IP address and port are correct.
2. The required script or FlyWithLua files have been copied.
3. The simulator was restarted after integration changes.
4. Parameters appear in the snapshot.

If the script is missing or deployed to the wrong place, rules will not work even if the profile itself is correct.

XPLANE 12: SETTINGS ARE FILLED IN BUT TELEMETRY IS STILL SILENT

Check these in order:

1. **XPlane 12** is selected on the Dashboard, not XPlane 11.
2. LinkTool runtime is **running** (START).
3. X-Plane 12 has **Accept incoming connections** enabled and a **flight** is loaded.
4. Integrations IP and **legacy RREF port** are correct (`127.0.0.1:49000` on one PC — not `49010`).
5. The local UDP port is free (try changing `49001` to `49003` if it conflicts with X-Plane).
6. The **X-Plane 12 Datarefs** page has enabled entries and **Save datarefs** was clicked.
7. Dataref paths are recognized by the loaded aircraft or plugin (built-in XP12 paths often use `sim/cockpit2/...` ; custom paths do not need a `sim/` prefix).

For XP12 you **do not** need FlyWithLua or files under `Resources\plugins` . If you configured XPlane 11 before, that is a different simulator entry.

WAR THUNDER: DATA ARRIVES INCONSISTENTLY

Check these in order:

1. The address and port are correct.
2. The **poll interval** and **request timeout** values are appropriate.
3. Parameters appear in the snapshot at least intermittently.

If values arrive in bursts or fail intermittently, the cause is often closer to the HTTP address, poll interval, or timeout setting than to the LED rules.

AUTOMOBILISTA 2: NO SHARED MEMORY DATA

Check these in order:

1. **Options** → **System** → **Use Shared Memory** is **Project CARS 2** and **Save** was clicked.
2. **UDP Protocol Version** is **Project CARS 2** (matches the shared memory mode AMS2 expects for external apps).
3. You are **on track** in a driving session, not in menus or replay.
4. **Automobilista 2** is selected in LinkTool and **Settings** → **Integrations** → **Shared Memory** was saved.
5. Map name is `$pcars2$` (default).
6. LinkTool was **started after** you loaded the session (restart LinkTool if you changed AMS2 options mid-session).

If the telemetry drawer stays empty but AMS2 is running, load a practice session, drive a few seconds, then stop and start LinkTool once.

ASSETTO CORSA EVO: NO SHARED MEMORY DATA

Check these in order:

1. AC Evo is **on track** in a driving session, not in menus or replay.
2. **Assetto Corsa Evo** is selected in LinkTool and **Settings** → **Integrations** → **Shared Memory** was saved.
3. LinkTool was **started after** you loaded the session (restart LinkTool if you changed settings mid-session).
4. LinkTool and AC Evo run under the same Windows user session.
5. After a **game patch**, verify LinkTool supports your build — check `AcEvoVersion` and `SmVersion` when telemetry is working.

If the drawer stays empty but the game is on track, drive a few seconds, then stop and start LinkTool once.

LE MANS ULTIMATE: NO SHARED MEMORY DATA

Check these in order:

1. LMU is **on track** in a driving session, not in menus or replay.
2. **Le Mans Ultimate** is selected in LinkTool and **Settings** → **Integrations** → **Shared Memory** was saved.
3. LinkTool was **started after** you loaded the session (restart LinkTool if you changed settings mid-session).
4. LinkTool and LMU run under the same Windows user session.
5. After a **game patch**, verify LinkTool supports your build.

If the drawer stays empty but the game is on track, drive a few seconds, then stop and start LinkTool once.

IRACING: NO SHARED MEMORY DATA

Check these in order:

1. iRacing is **on track** in a live driving session (in the car), not in menus or replay.
2. **iRacing** is selected in LinkTool and **Settings** → **Integrations** → **Shared Memory** was saved.
3. LinkTool was **started after** you loaded the session (restart LinkTool if you changed settings mid-session).
4. LinkTool and iRacing run under the same Windows user session.
5. After an **iRacing update**, verify LinkTool supports your build.

If the drawer stays empty but the game is on track, drive a few seconds, then stop and start LinkTool once.

PROJECT MOTOR RACING: NO UDP TELEMETRY

Check these in order:

1. PMR **Settings → Preferences → UDP Enabled** is **On**.
2. **UDP Host** is `127.0.0.1` and **UDP Port** matches LinkTool (default `7576`).
3. You are **on track** in a driving session.
4. **Project Motor Racing** is selected in LinkTool and **Settings → Integrations → Project Motor Racing** was saved.
5. Another PMR telemetry tool is **not** already using the same UDP port — close it or enable multicast in both PMR and LinkTool.
6. LinkTool was **started after** you enabled UDP or changed ports (restart LinkTool once after game-side changes).
7. Windows Firewall allows LinkTool to receive UDP on the configured port.

If values appear briefly and then stop, check for a port conflict first.

UNIVERSAL ISOLATION ORDER

If you are not sure where the failure lives, always narrow it down in this order:

1. device;
2. selected simulator;
3. simulator connection settings;
4. telemetry snapshot;
5. one simple rule;
6. only then a larger profile.

> Screenshot placeholder SC-09: Diagnostics view or system log area.

FAQ SECTION

SECTION 16 FAQ guidance and operational notes for VIRPIL Controls LinkTool.

FAQ

DO I ALWAYS NEED A NEW PROFILE

No. If you already have a working profile, you can open and extend it. `New profile` is mainly useful for a clean start or a controlled test scenario.

WHICH IS BETTER: AUTO-SAVE OR MANUAL SAVE

If you make frequent small edits and want the current profile to stay up to date automatically, auto-save is convenient. If you prefer tighter control and often experiment with rules, manual save may feel safer.

WHY SHOULD I TROUBLESHOOT ONE SIMULATOR AT A TIME

Because the fastest way to isolate a problem is to think about one simulator at a time. If you jump between multiple simulator setups during configuration, it becomes much harder to tell whether the failure is in the connection, the incoming data, or the rule itself.

In practice, pick one simulator, save its settings, verify its telemetry snapshot, and only then move on.

WHY IS IT BETTER TO TEST ONE RULE FIRST

Because it makes failures easier to isolate. If you add many rules at once, it becomes much harder to tell which rule is misconfigured or interfering.

CAN I TEST THE DEVICE BEFORE USING A SIMULATOR

Yes. That is exactly what manual LED control is useful for. It lets you confirm that the basic connection to the device works before bringing telemetry into the picture.

DOES MY CHATGPT OR CLAUDE PHONE SUBSCRIPTION WORK AS AN API KEY

No. ChatGPT **Go**, Plus, Pro, and Claude chat subscriptions are for the consumer apps. LinkTool uses **developer API keys** from platform.openai.com (ChatGPT tab) or console.anthropic.com (Claude tab), with separate pay-as-you-go billing.

For free telemetry explanations, use **Groq** as the active provider and create a key at console.groq.com.

X-PLANE 11 OR X-PLANE 12 — WHICH ONE DO I SELECT

- **X-Plane 12** → choose **XPlane 12** on the Dashboard. No plugins are required: configure **Integrations**, your dataref list, then start LinkTool. See the **XPlane 12** section in this README.
- **X-Plane 11** → choose **XPlane 11**, deploy FlyWithLua and the LinkTool script. That is a different integration path (the game pushes JSON to LinkTool).

You only need one active X-Plane entry on the Dashboard at a time.

Adjust the polling settings before rewriting the rules.

- For local input-driven behavior, start with `Input polling interval (ms)`.
- For the SimConnect family, adjust the dedicated SimConnect polling interval.

If the data stream itself is unstable or too dense, rule tuning becomes guesswork.

WHERE DO I CHANGE THE LANGUAGE AND THEME

Use the controls in the top header of the main window.

- the language combo box switches the UI language;
- the theme button switches appearance;
- neither action changes the profile logic itself.

These are application-level preferences, not rule settings.

WHERE IS THE BUG REPORT WINDOW

Use the `Report issue` button in the top header of the main window.

Before opening it, it is often worth checking Diagnostics first so you already know:

- whether the problem is reproducible;
- where the logs are;
- which log file is the newest;
- whether you need to export a logs archive.

WHERE IS THE PDF MANUAL IN THE APP

Use the `PDF` button in the top header.

It opens the bundled documentation file that ships with the application. The preferred language depends on the current UI language.

WHY DO ALL LEDS TURN OFF OR CHANGE COLOR WHEN I PRESS START

Check **Settings** → **General** first:

- **Reset LEDs before start** controls whether LinkTool sends an explicit reset on **START** and **STOP**;
- **LEDs without rules** controls LEDs that have no rule in the active profile while LinkTool is running.

If you only want LinkTool to manage LEDs that have explicit rules, set **LEDs without rules** to **Firmware defaults** and disable **Reset LEDs before start** if you do not need a full reset at runtime start.

WHY DOES RESET LEDS BEFORE START SEEM TO DO NOTHING

This usually happens with **LEDs without rules** = **Firmware defaults**:

- the toggle applies only when you press **START** or **STOP**, not while LinkTool is already running;
- if the panel already shows firmware colors, **On** (command `100`) and **Off** (leave unchanged) often look identical on **START**;
- the difference is easier to see on **STOP** while a rule is still active, or after manual LED testing.

See the **Reset LEDs Before Start** section in Settings documentation for the full explanation and test scenarios.

WHY CAN IL-2 GREAT BATTLES LOOK BROKEN EVEN WHEN THE RULE IS FINE

One common cause is a UDP conflict with another telemetry tool. If the game, LinkTool, and another utility are all trying to use the same data flow, the telemetry snapshot can look empty, partial, or mixed.

When that happens, check the connection settings first and only then go back to the rules.

SECTION 17

For future PDF work, it helps to keep every screenshot block in the same format.

SCREENSHOT PLACEHOLDERS FOR FUTURE PDF WORK

Recommended screenshot template:

- ID: `SC-XX`
- File name: short, Latin characters, hyphen-separated
- Caption: one short sentence describing what the user sees
- Purpose: one line explaining what the image should teach

Recommended set:

- `SC-01` — Startup dashboard File name: `01-startup-dashboard.png` Caption: Main dashboard after LinkTool startup and device detection. Purpose: show the user's starting point and the primary actions.
- `SC-02` — Dashboard zones File name: `02-dashboard-zones.png` Caption: Main working areas of the Dashboard. Purpose: explain the overall interface structure quickly.
- `SC-03` — Simulator and telemetry File name: `03-simulator-telemetry.png` Caption: Simulator selection and the Telemetry snapshot drawer. Purpose: show where incoming data is checked.
- `SC-03A` — Telemetry lab File name: `03a-telemetry-lab.png` Caption: Telemetry lab with search, groups, watchlist, and create-rule actions. Purpose: show the full telemetry explorer used for rule testing.
- `SC-03B` — Telemetry EXPLAIN dialog File name: `03b-telemetry-explain.png` Caption: AI explanation dialog with parameter name, simulator context, and cached-answer badge. Purpose: show optional AI help while exploring telemetry parameters.
- `SC-04` — Profile actions File name: `04-profile-actions.png` Caption: Creating, opening, and saving a profile. Purpose: explain the profile file workflow.
- `SC-05` — Rules list File name: `05-rules-list.png` Caption: Compact rules list on the Dashboard. Purpose: show where the active ruleset lives.
- `SC-06` — Rule editor File name: `06-rule-editor.png` Caption: Overlay editor for a single rule. Purpose: explain how a rule is configured.
- `SC-06B` — Duplicate rule wizard File name: `06b-duplicate-rule-wizard.png` Caption: Duplicate rule wizard on the Review step with LED summary and conflict warning. Purpose: show how to fan out gradient or telemetry rules to other LEDs.
- `SC-07` — LED bench File name: `07-led-bench.png` Caption: Manual LED control area. Purpose: show the fastest hardware validation path.
- `SC-08` — Settings File name: `08-settings-tabs.png` Caption: Settings page with integration and general configuration areas. Purpose: show where the operating environment is configured.
- `SC-08A` — AI Integrations File name: `08a-ai-integrations.png` Caption: Settings → AI Integrations with enable toggle, active provider, and API key fields. Purpose: show how to configure Groq, ChatGPT, or Claude for EXPLAIN.
- `SC-09` — Diagnostics File name: `09-diagnostics.png` Caption: Diagnostics, status, and system messages. Purpose: show where users should look when something fails.

If you turn this into a PDF later, these same IDs and captions can be reused directly as figure titles with almost no extra formatting work.

RECOMMENDED WORKING STYLE

SECTION 18

The most practical LinkTool workflow usually looks like this:

RECOMMENDED WORKING STYLE

1. Verify device connection.
2. Verify telemetry.
3. Create one rule.
4. Test the result.
5. Save the profile.
6. Only then make the logic more complex.

This is usually much faster than trying to build a large profile all at once and troubleshooting only after everything has become difficult to untangle.