LogRunner

User Manual

Version	**0.9.3b61** (Beta)
Date	2026-09-14
Language	English

Contents

1. Overview & Purpose

LogRunner is a portable offline logger for amateur radio. It keeps the station log in a single SQLite file and works entirely without an internet connection. All online services – Wavelog sync, DX cluster, callsign lookup, band conditions – are extras, are **switched off** out of the box and can be enabled one by one.

The program is meant for radio amateurs who want to keep their log in their own hands: for everyday operation in the shack, for portable operation without a network, for contest weekends and for stations that run several screens or several PCs side by side.

All data lives in **one** folder (`~/LogRunner/`). That folder can be copied to a USB stick and used on another computer – back it up and you have backed up everything.

Main features at a glance

- **QSO entry** through an entry mask with seven tabs (QSO, Station, General, Contest, Sat, Notes, QSL) – the same field layout as in Wavelog, with ADIF fields throughout.
- **Live callsign resolution** via `cty.dat`: country, continent, CQ and ITU zone and the DXCC number are filled in automatically when logging.
- **Log table** with search and filter, freely selectable columns (81 to choose from, grouped by the tabs of the entry mask), sorting by any column and edit/delete per QSO.
- **Worked bands**: band matrix per callsign with the states *worked / confirmed / not yet*, optionally broken down by mode.
- **Statistics**: tiles, activity over time, band-versus-mode matrix, breakdown by continent, year, location and operator, plus IOTA progress.
- **Offline world map** in three views (flat map, globe, azimuthal map) with grey line, DXCC prefixes, CQ/ITU zones, locator grid, place names, IOTA island groups, DX spots and log entries.
- **DX cluster** via Telnet with table, vertical band map and spot statistics, wildcard filters, saved filter sets and audible alarm rules.
- **Contest operation** with session, serial numbers, dupe warning, a keyboard-driven contest line and Cabrillo submission.
- **TRX control** of the transceiver via Icom CI-V, Hamlib (`rigctld`) or Omnirig – including a waterfall from the band scope of Icom rigs.
- **WSJT-X link**: whatever is logged in WSJT-X appears in the log seconds later.
- **Rig screen via an HDMI capture card**, optionally offered on the network.
- **ADIF import and export** with a tolerant parser and duplicate handling.
- **Wavelog/Cloudlog sync** via API v1 and v2 with an offline queue.
- **LiveLink** of several PCs on the LAN (one hub, any number of clients), where each computer can show a single area full-screen.
- **POTA, SOTA and IOTA lists** searchable offline, plus the Super Check Partial list for callsign suggestions.
- **Six languages** (Deutsch, English, Español, Français, Italiano, Português), switchable without a restart, and a light and a dark theme.

2. System Requirements & Installation

2.1 Requirements

Requirement	Details
Operating system	Windows 10/11 (ready-made beta). macOS and Linux run from source.
Display engine	**Microsoft Edge WebView2 Runtime**, at least Chromium version 111
Python (source only)	**3.11 or newer**
Node.js (only to build the user interface)	current LTS release
Disk space	about 50 MB for the program, plus the data folder

> ⚠️ **Caution:** LogRunner does not start without the WebView2 runtime. The program detects this and says so in a message box; it deliberately does **not** silently fall back to the old Internet Explorer engine, in which the user interface would stay black. Remedy: install the *Evergreen Standalone Installer (x64)* for WebView2 from Microsoft and restart LogRunner.

2.2 Dependencies (from `pyproject.toml`)

Package	Minimum version	Used for
`pywebview`	5.1	The window and the bridge between user interface and Python
`tomli-w`	1.0	Writing `config.toml`
`pyserial`	3.5	Serial line to the Icom CI-V bus (loaded only when needed)
`websockets`	12.0	LiveLink between several PCs
`zeroconf`	0.131.0	Finding the hub automatically via mDNS

Optional:

Package	Used for
`comtypes` ≥ 1.2 (`pip install logrunner[omnirig]`)	Only for the Omnirig connection, only on Windows
`pytest` ≥ 8.0, `ruff` ≥ 0.16, `mypy` ≥ 1.11, `pyyaml` ≥ 6.0, `pyinstaller` ≥ 6.11, `pillow` ≥ 10.0 (`[dev]`)	Tests, checking tools and release builds

External programs that LogRunner does **not** bring along:

- **ffmpeg** – only for screen capture through a capture card. A button in the *Screen capture* panel downloads it by itself on Windows (about 110 MB, once).
- **rigctld** (Hamlib) or **Omnirig** – only for the corresponding CAT connection.

2.3 Installing the Windows beta

1. Unpack the archive `LogRunner-<version>-win64.zip` **completely** – the folder contains `LogRunner.exe` and, next to it, the directory `_internal`.
2. Put the unpacked folder anywhere you like (hard disk, USB stick, network drive). There is no installation as such, and no registry entries are written.
3. Start `LogRunner.exe` – by double-click or through a shortcut.

> ⚠️ **Caution:** The program is a folder, not a single file. If only `LogRunner.exe` is copied out, the web part is missing and startup aborts with the message "*The web part of the program is missing*".

2.4 LogRunner on the Raspberry Pi

Tested on a **Raspberry Pi 5 (8 GB) with Raspberry Pi OS / Debian 13 (trixie), 64 bit**. The archive is called `LogRunner-<version>-linux-arm64.zip`; it runs on 64-bit ARM only, not on a PC.

1. Unpack – preferably where the archive is, for example in `~/Downloads`. As on Windows: the whole archive, not just the program file.

```
cd ~/Downloads
unzip LogRunner-0.9.3b61-linux-arm64.zip
```

2. Install – with the script from the archive:

```
bash LogRunner/install.sh
```

The script does in one go what would otherwise be done by hand (see below), and only asks where it has to decide something:

Step	What happens
Check	Right computer (64-bit ARM), not started as root, **no overlay file system** – if it is on, the script stops and explains how to switch it off, because otherwise everything would live in RAM only and be gone after the next reboot, log included
System packages	WebKitGTK (displays the user interface) and gio (starts the companion windows) are installed if missing. Hamlib and ffmpeg only if you answer yes when asked. This needs the `sudo` password
Transceiver	Your user is added to the `dialout` group so LogRunner may open the serial port. Log out and back in once; the script reminds you
Program	Copied to `~/Programme/LogRunner`. If a version is already there it is **moved aside**, not deleted; the last two moved-aside copies are kept, older ones are cleaned up by the script
Start menu	An entry under *Hamradio* ▸ *LogRunner* – to start it or to pin it to the panel

An update works the same way: close all LogRunner windows (the hub takes its companion windows with it), unpack the new archive, run `bash LogRunner/install.sh`. If LogRunner is still running, the script stops instead of swapping the program underneath the open windows. The **data folder** `~/LogRunner` – log, settings, log files – is never touched.

`bash LogRunner/install.sh --help` shows further options: a different location (`--prefix`), no questions (`--yes`), no system packages (`--no-apt`) or no cleanup (`--keep-all`).

By hand

If you prefer to do it yourself, you need the same steps one by one.

Display library and gio – LogRunner brings GTK along, WebKitGTK comes from the system:

```
sudo apt install gir1.2-webkit2-4.1 libglib2.0-bin
```

If you drive the transceiver through Hamlib instead of CI-V, add `libhamlib-utils`; for screen capture add `ffmpeg`. If WebKitGTK is missing, LogRunner itself says at startup which package to install.

Put the program in place – move the unpacked folder:

```
mkdir -p ~/Programme
mv ~/Downloads/LogRunner ~/Programme/
chmod +x ~/Programme/LogRunner/LogRunner
```

> ⚠️ **Do not unpack or move it to `~/LogRunner`.** That is exactly where the **data folder** lives – log, `config.toml`, log files. Program and data would otherwise end up in the same directory, and an update that replaces the program folder would take the log with it. `~/Programme` is only a suggestion; any other place works too – just not this one.
>
> **Window positions under Wayland.** The Raspberry Pi desktop (labwc) is a Wayland compositor, and there no window can remember its position – LogRunner neither; size and view it can. There are still two ways: a window rule of the compositor, or starting LogRunner with `GDK_BACKEND=x11`. Both are described under *Remember as startup state*.

3. Start – by double-click in the file manager or on the console:

```
~/Programme/LogRunner/LogRunner
```

A start menu icon is created by this file – write it once, and LogRunner appears in the menu under *Hamradio*, from where it can be started or pinned to the panel:

```
cat > ~/.local/share/applications/logrunner.desktop <<'ENDE'
[Desktop Entry]
Type=Application
Name=LogRunner
Comment=Offline log for amateur radio
Exec=/home/USER/Programme/LogRunner/LogRunner
Path=/home/USER/Programme/LogRunner
Icon=/home/USER/Programme/LogRunner/_internal/logrunner/assets/logo.png
Terminal=false
Categories=HamRadio;Network;
ENDE
```

Replace `USER` with your own user name – the lines must carry absolute paths; a `.desktop` file does not understand `~`.

The transceiver shows up as `/dev/ttyACM0`, not as `/dev/ttyUSB0` – an IC-7300MK2 is a USB composite device and comes through the CDC-ACM driver. Looking for `ttyUSB` there is looking in the wrong place. Your user must be in the `dialout` group:

```
sudo usermod -aG dialout $USER
```

Then log out and back in once.

What is different on a Pi compared with Windows:

Display	WebKitGTK instead of WebView2
Shortcuts for companion windows	`.desktop` files instead of `.lnk`
Transceiver connection	CI-V and Hamlib; **Omnirig** does not exist there

Serial ports	`/dev/ttyACM*` and `/dev/ttyUSB*`

Measured on a Pi 5 with an IC-7300MK2 attached: a status query of the rig takes 3.7 ms, the full control bar 33 ms, the waterfall runs at 3.3 sweeps per second without a single dropout – the pace is set by the transceiver, not the Pi. Only **zooming the world map** is noticeably slower: one notch of the mouse wheel costs about half a second there, against a tenth of a second on Windows. Panning costs the same on both.

2.5 Running from source

```
python -m venv .venv
.venv/Scripts/pip install -e ".[dev]"
```

Build the user interface once:

```
npm --prefix frontend install
npm --prefix frontend run build
```

Then start:

```
.venv/Scripts/python -m logrunner
```

Alternatively, the command `logrunner` is available after installation. If there is no frontend build, the window connects to the Vite development server at `http://localhost:5173`.

A Windows beta is built with:

```
.venv/Scripts/python tools/build_release.py
```

2.6 The data folder

On first start LogRunner creates its data directory. The default is `~/LogRunner/` (on Windows `C:\Users\<name>\LogRunner\`). The environment variable **`LOGRUNNER_HOME`** overrides this location – that way the log can live on a stick, for example.

```
~/LogRunner/
├── logrunner.sqlite      The log itself
├── config.toml           All settings
├── license.json          Record of the accepted licence
├── .logrunner.lock       Lock against a second start
├── data/
│   ├── cty.dat           Country prefix file (DXCC resolution)
│   ├── solar.json        Last fetched solar activity figures
│   ├── muf.json          Last fetched MUF contours and station values
│   ├── dxcluster.sqlite  Spot history of the DX cluster
│   ├── propagation.sqlite Readings for the trend display
│   └── webview/          Cookies/logins of the display library
├── backups/              Automatic backups (the last 20)
├── exports/              Written ADIF and Cabrillo files
├── logs/                 logrunner.log, start.log
├── shortcuts/            Shortcuts of the companion windows
└── clients/              Data directories of the companion windows
```

2.7 The start window and the licence

Before the main window a small start window appears with the logo, the version number and a note about the end-user licence agreement. In normal operation it is only there for the second or two the main window needs to build up, and then disappears by itself.

On the very first start and after every new version, however, it is a question: it shows the full licence text and waits for an answer.

- **Accept and start** – first tick *I have read and accept the licence terms*, then the button. The main window opens afterwards.
- **Cancel** (or the close button of the title bar, or `Esc`) – the program quits without showing the main window. Nothing is sent to the network, no radio is addressed and no sync is started.

The consent is recorded in `~/LogRunner/license.json` – with the version and a fingerprint of the licence text. You are therefore asked again as soon as a new program version runs **or** the licence text itself has changed. The record applies to the whole installation: a companion window (`--client`) does not ask again.

The licence text lies next to `LogRunner.exe` as `LICENSE.txt`, the terms of the included third-party components as `THIRD-PARTY-NOTICES.txt` beside it. Both can be read at any time without the program running; the start window names their paths.

> The licence agreement is binding in German only. The labels of the start window follow the language set in `config.toml`; in the other five languages a corresponding note is shown.

3. User Interface & Navigation

3.1 The main window

The window is divided into three horizontal areas: the **header** with all menus, the **work area** with entry mask and tab area, and the **status bar** at the bottom edge.

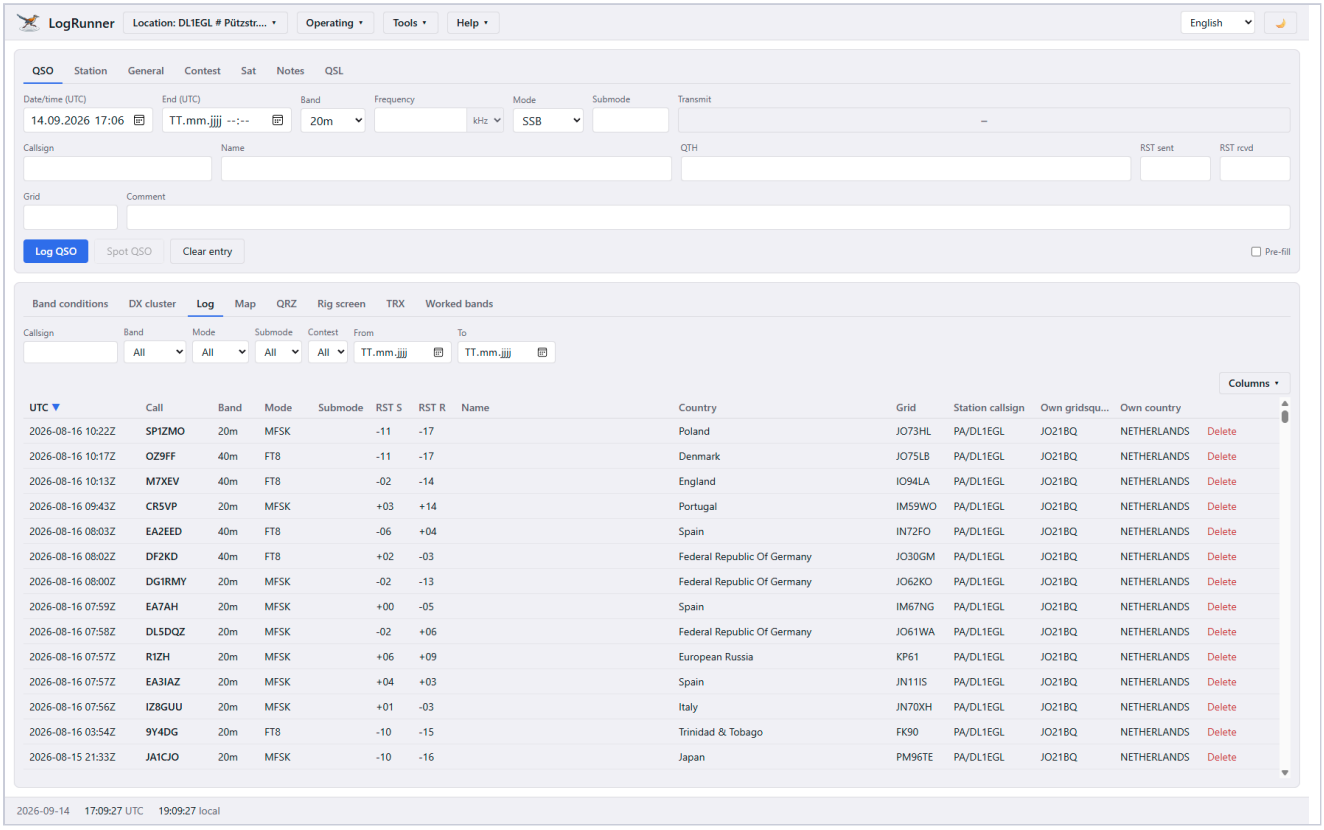

Fig. 1: The main window in the full view with entry mask, tab bar and log table

> The pictures in this manual show the **light theme**. With the dark theme you see the same controls in the same places – the ☀ / 🌙 switch sits at the top right of the header.

Header (menu bar)

From left to right:

Element	Meaning
LogRunner	Program name
Location ▾	The station location the next QSO is logged under. The button names the active location with its callsign; a tick marks it in the list.
Operating ▾	*Logbook* (full mask) or *Contest* (the short contest line).
Preset ▾	The window combinations of the LiveLink. A click brings the preset up and closes the companion windows that do not belong to it. The button names the preset that is running; the menu lists the names, one of them ticked. **Only shown when LiveLink is switched on and this window runs the hub** – a client has no presets of its own and could not start them either. They are created under *Tools* ▸ *LiveLink* ▸ *Start companion windows too*.

Element	Meaning
Tools ▾	All settings and tool dialogs (see below).
Help ▾	*About* – version, data paths, system information, licences.

At the right edge of the header:

Element	Meaning
"n waiting"	Only appears when QSOs are waiting for the Wavelog upload. A click opens the sync dialog.
Language	Deutsch, English, Español, Français, Italiano, Português – takes effect at once, without a restart.
☀ / 🌙	Switch between the light and the dark theme.

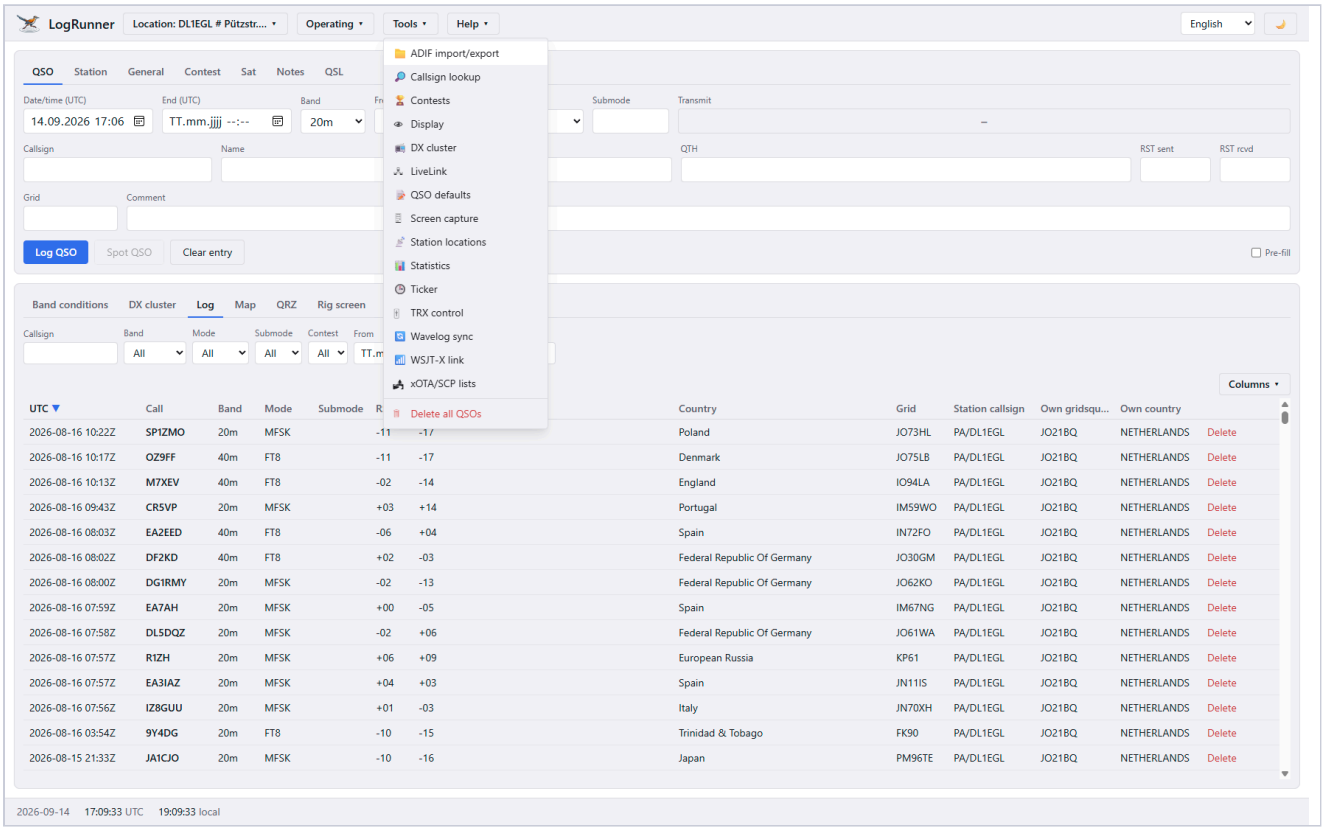

Fig. 2: The header with the menus Location, Operating, Tools and Help – the Tools menu opened

> **Note on *Preset* ▾:** The button names the preset that **this program run** brought up – by autostart or by hand. As long as none has been started it just says *Preset*, even if one is chosen for autostart in the dialog: that applies to the next start and says nothing about which windows are up right now. If a preset cannot be started, the button gets a red border for a few seconds, and the reason appears in the tooltip and at the top of the opened menu.
>
> **Since 01.09.2026 the *View* is no longer here but in the right-click menu** (section 3.2). The reason: what can be chosen depends on what the window shows at all, and a menu in the header is already there before anyone clicks on it.

If a menu does not fit into the window, it scrolls. This mainly affects *Tools* in a low window: instead of being cut off at the bottom, the list gets a scroll bar. The mouse wheel stays in the menu and does not take the area underneath along.

The whole header can be switched off, and the way there is a **right-click** anywhere in the window: *Menu bar off*, and when it is gone, the same place offers *Menu bar on*. This works in every view, including the compact one, and also

over an input field.

It is the only way, and deliberately so: the Tools menu disappears with the header, so a tick in it would only be reachable for as long as you did not need it. The setting is meant for a companion window that only shows a map or the rig screen and needs none of these menus.

The *Tools* menu

The entries are sorted alphabetically by their translated name, so they sort differently in every language. Only *Delete all QSOs* is the exception: it sits at the bottom behind a separator, in red.

Entry	Purpose
📁 ADIF import/export	Read QSOs from `.adi` files and write them out
🔎 Callsign lookup	QRZ.com, HamQTH, callook.info
🏆 Contests	Contest list and Cabrillo submission
👁 Display	How QRZ profiles open, IARU band plan region
📻 DX cluster	Nodes, login callsign, buffer size, skimmer – on a client the tick *Follow the hub*
🖧 LiveLink	Hub/client, mask, companion window shortcuts
📝 QSO defaults	Named sets of start values for the mask
🖥 Screen capture	Set up a capture card, offer the picture on the network
📡 Station locations	Create your own locations with their master data
📊 Statistics	Evaluations of the log
🕒 Ticker	Station clock, callsign and QSO counter for a screen of its own
🎚 TRX control	Connection to the transceiver
🔄 Wavelog sync	Address, key, upload and fetch
📶 WSJT-X link	UDP reception from WSJT-X
🏕 xOTA/SCP lists	Fetch POTA, SOTA, IOTA and SCP lists
🗑 **Delete all QSOs**	**Destructive** – see the warning in section 4.9

Each entry opens a dialog above the work area. **Esc** or the *Close* button closes it; the cursor then jumps back into the callsign field.

The entry mask

Seven tabs sit above the mask. A tab in which something is entered is marked – so nothing an import or a lookup brought along stays hidden.

Tab	Content
QSO	The contact itself – see the row layout below
Station	Receive band and frequency (cross-band, satellite), operator, TX power
General	DXCC, country, continent, region, CQ/ITU zone, state, county, propagation mode, antenna path, e-mail, IOTA, SOTA, WWFF, POTA, SIG, DOK
Contest	Contest ID, both serial numbers, both exchange fields
Sat	Satellite name, satellite mode, antenna azimuth and elevation
Notes	Multi-line QSO note (ADIF `NOTES`)

Tab	Content
QSL	Six ways, each with *Sent* and *Received*: QSL card, eQSL, LoTW, QRZ, Clublog, DCL – see below

The **QSO** tab is arranged in three rows:

> **Mode and submode.** The selection shows the nine you touch in operation at the top, below them *more ADIF modes* with all the others. *FT4*, *JS8* and *PSK31* are submodes in ADIF – you pick the familiar word, the pair goes into the log (FT4 becomes `MFSK` with submode `FT4`). The band filter finds a QSO under either.

1. **When and with what:** Date/time (UTC) · End (UTC) · Band · Frequency · Mode · Submode · *Transmit* (the on-air indicator).
2. **The other station:** Callsign · Name · QTH · RST sent · RST rcvd.
3. **Additions:** Grid · Comment.

Below that the button bar: **Log QSO** · **Spot QSO** · **Clear entry** · (during a contest) **Contest line / Full mask**. At the far right sits the tick **Pre-fill**. Between the buttons, badges appear when needed: *worked: n QSOs on m bands*, the distance with bearing, the running contest number and the red dupe warning.

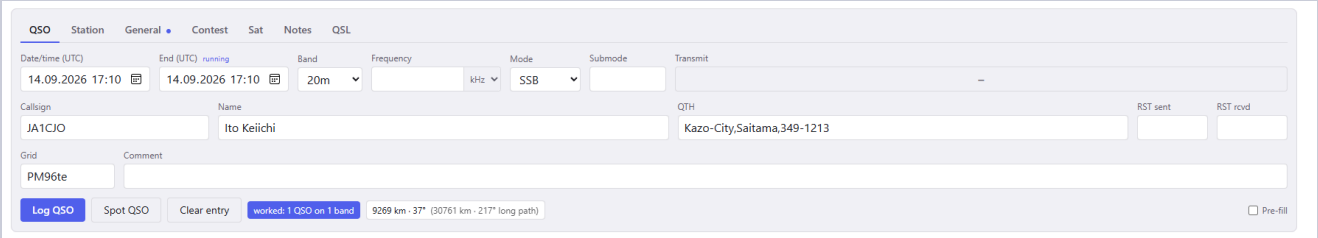

Fig. 3: The entry mask in the QSO tab with the three field rows and the button bar

The *QSL* tab

Six ways to send or receive a QSL, and each has a section of its own with a heading:

Section	Fields
QSL card	Sent, send method, sent via, received, receive method, QSL message sent, QSL message received
eQSL	Sent, received
LoTW	Sent, received
QRZ	Sent, received
Clublog	sent only – ADIF knows no receive direction for Club Log
DCL	Sent, received

The five electronic ways stand **side by side**, one per column, with their two fields below. From about 1000 px window width all five fit in one row; if the window gets narrower, they wrap to three and then to two columns.

The two **message fields** belong to the card and not to eQSL: ADIF defines `QSLMSG` and `QSLMSG_RCVD` as the message of the *card*, and the standard has no eQSL message of its own.

The states are **ADIF letters, not tick boxes** – *Requested* and *Verified* thus survive an edit instead of collapsing to yes/no. QRZ and Clublog have one state more than the others: *Modified – upload pending*.

Two states are **no longer offered** by the mask, because ADIF only intends them for *import*: *Verified* under "Received" and *Manager* for both ways. A QSO that brought them along from an import keeps them nonetheless and shows them.

> **Empty is not the same as No.** For eQSL, LoTW, QRZ, Clublog and DCL the default is — *no information* —. LogRunner itself uploads nothing to any of these services; a *No* on every QSO would be a statement about something nobody ever asked. *No* means **not sent**, empty means **nothing known**. Only *LoTW received* has no empty state – that field has existed with *No* since the first version.

The date fields are not in the mask. For each of the six ways ADIF has a date for sending and one for receiving; they come in with an **ADIF import** (from Wavelog, from QRZ.com, from Club Log) and can be shown as **columns of the log table** – under *Columns* ▸, in the group *QSL*. Entering them by hand would be a line of typing for something another program decides anyway. Editing in the mask leaves them untouched.

The tab area

Below the mask lies a tab bar. It too is sorted alphabetically by the translated names.

Tab	Content
Band conditions	Solar flux, sunspots, A/K index, MUF, X-ray flux, noise level and the rating per band group, plus trend arrows, the path indicator for one path and the band indicator for the continents
DX cluster	Spots as table, band map or band activity *(only with the cluster switched on)*
Log	Filter row and log table *(selected at startup)*
Map	World map, globe or azimuthal map
QRZ	What the callsign services know about the station in the mask
Rig screen	The transceiver's screen via the capture card
TRX	Spectrum from the band scope and the rig's control bar *(only when switched on)*
Worked bands	Band matrix for the callsign in the mask

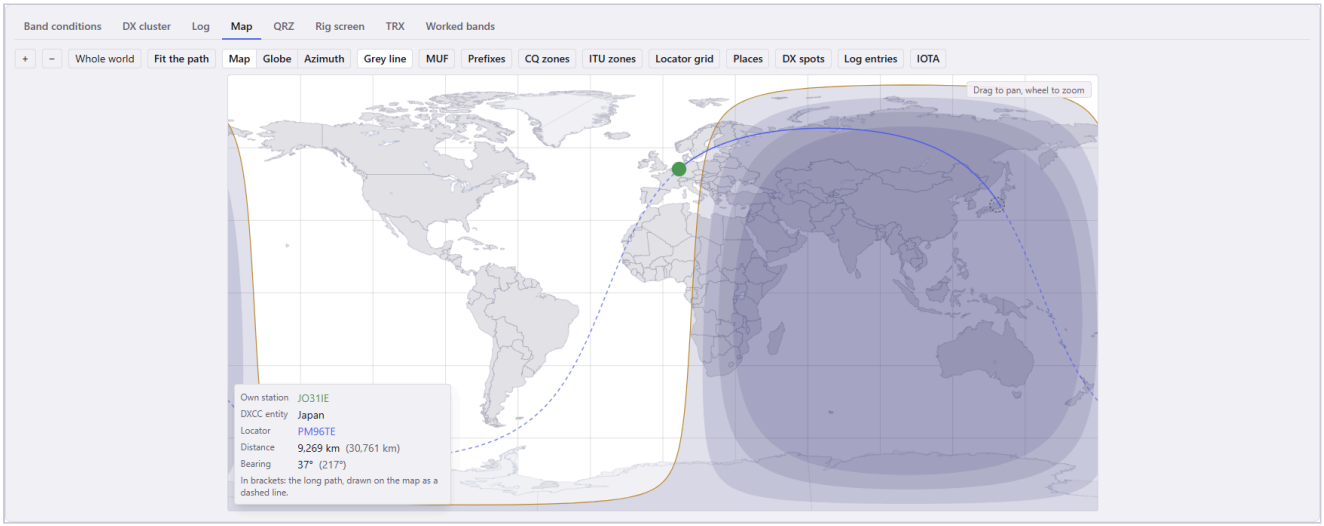

Fig. 4: The tab area with the world map open

The status bar

Permanently shown at the bottom: **date**, **UTC time** and **local time**. When the respective function is switched on, indicator lamps are added – the state of the WSJT-X link, of TRX control (with a red **On Air** while transmitting) and of LiveLink (with role, **name of this window** and number of connected PCs).

The name is the one under which the window appears in the PC list and in the band assignment – that is, the one assigned by the hub, which carries a digit if the desired name was already taken. It is shown because the role alone

only half answers the question: if several companion windows of the same kind are open – two transceiver windows, say – "Client" says the same on each of them.

The bar can be switched off under *Tools* ▸ *Display*.

3.2 The view

It is in the right-click menu, anywhere in the window – not in the header. What can be chosen depends on what this window shows (mask, section 3.1 and *Tools* ▸ *LiveLink*):

The window shows	Choices
Everything (whole window)	*Compact · Full*
TRX (`--mask scope`)	*Waterfall · Rig controls · Full*
Band conditions (`--mask solar`)	*Solar activity · Path indicator · Band indicator · Detail*
Map, Log, DX cluster, Rig screen, …	– nothing; there is only this one area there

A tick shows which one applies. The choice applies to *this* window only, is stored in its `config.toml` and survives a restart.

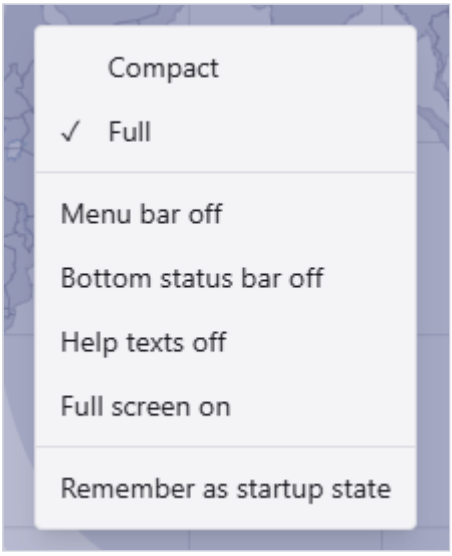

Fig. 5: The right-click menu in the whole window: Compact and Full, below them menu bar, status bar, help texts, full screen and remember as startup state

Below the view, the same menu holds the switches for the **menu bar**, the **bottom status bar**, the **help texts** and **full screen**, and at the very bottom *Remember as startup state* (section 4.2). Over text, *Select all* is offered at the top as well.

Everything (whole window)

- **Full** shows the whole area and adapts so that everything fits into the window height without scrolling.
- **Compact** leaves only the entry mask and shrinks the window to its size – for contest operation or for a window parked in the corner next to a waterfall.

Both set the window to the width at which the mask stands as it is designed.

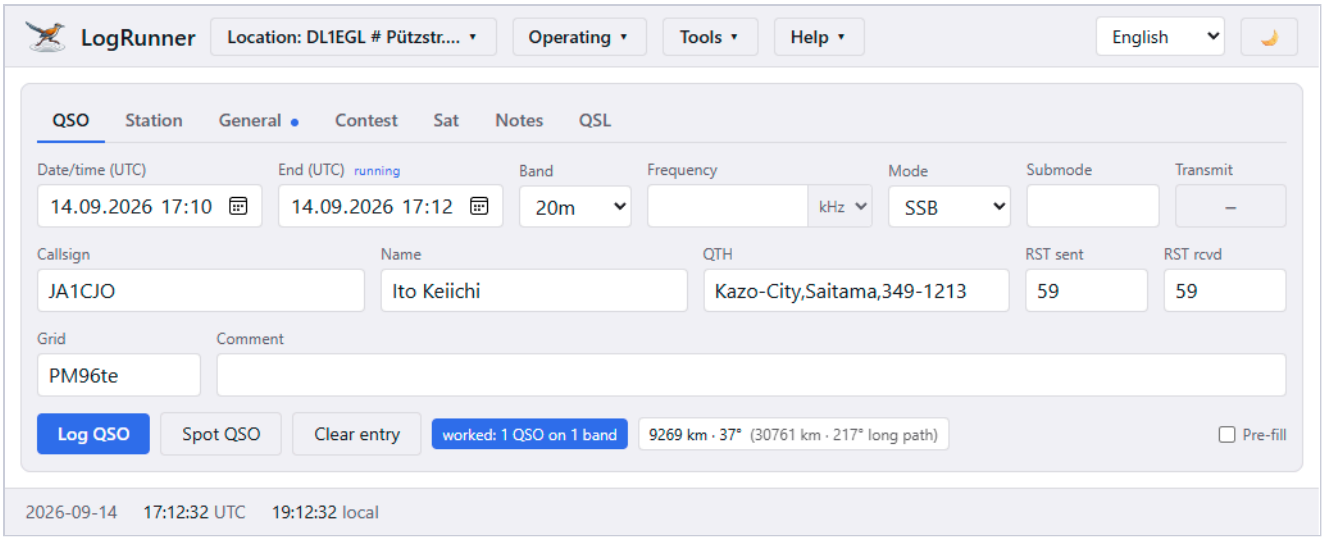

Fig. 6: The compact view: only the entry mask

TRX

On a computer that shows only the TRX area there is no entry mask that could be left out – so the question goes one level deeper and concerns the two halves of this area:

- **Waterfall** – the display only: curve, waterfall and the DX spots above it. A pure waterfall window for the screen next door.
- **Rig controls** – only the control bar, expanded and without the fold button in front. While it stands alone, LogRunner **no longer** asks the rig for spectra: the sweeps run over the same CI-V line as the controls, ten times a second, and whoever picks this view has ordered the line for the controls.
- **Full** – both, as the *TRX* tab looks in the full window.

> The waterfall does not keep its history when you put it away: the history is the drawing itself. After *Rig controls* and back, the picture starts anew.

Band conditions

The same one level deeper, on a computer that shows only this area. It consists of three parts, and each of them is worth a window of its own:

- **Solar activity** – the figures from the feed: the eight cards with their sparklines, the band table for day and night and the VHF row. With them the header holding *Refresh* and the switch.
- **Path indicator** – only the one path: the input field for callsign or prefix and the row of bands below.
- **Band indicator** – only the continent grid, which is tied to no callsign. The view for the second screen: *where can I get to at all right now*.
- **Detail** – everything, as the *Band conditions* tab looks in the full window.

After each choice the window measures its height again and shrinks to what is still there.

> The source line belongs to the figures and is therefore shown only with *Solar activity* and *Detail*: it names whose table this is. Where the two indicators get their data from is stated by the legend of the MUF layer on the map.

4. Features & Workflows

4.1 Workflow 1 – Logging a QSO

This is the program's main function.

Preparation (once). Under *Tools* ▸ *Station locations* create at least one location: label, own callsign, locator, QTH and whatever else belongs to it. The button *Complete from callsign* fills in country and zones from `cty.dat`. The location marked with ★ is active.

> ⚠️ **Caution:** The location data are copied into **every** QSO when logging – as they stand at the time of the QSO. Moving later and changing the location does not retroactively change where old QSOs were made. This is intentional.

Sequence per contact:

1. **Type the callsign.** From the first character on, several things happen at once: * A suggestion list opens with callsigns from your own log (highlighted) and from the Super Check Partial list. * The log table below filters itself to what you typed. * The clock for *End (UTC)* starts running. * If the callsign is already in the log, the badge "*worked: n QSOs on m bands*" appears. * A **double-click** in the field opens the QRZ.com profile.
2. **Set band, frequency and mode** – by hand, from a clicked DX spot or automatically from the transceiver when TRX control is running.
3. **Enter reports, name, QTH, grid, comment** as available. As soon as a locator is there and your own location has one, distance and bearing appear – short path, the long path in brackets.
4. Press **Log QSO** (or Enter). This happens automatically: * Country, continent, CQ/ITU zone and the DXCC number are added from `cty.dat` if the fields are empty. * The master data of the active location are stamped on. * If a contest is running, the QSO gets its ID, the next serial number and the fixed exchange. * The QSO is put into the Wavelog queue – regardless of whether the sync is set up at all. * The mask is cleared; with *Pre-fill* ticked, band, mode, submode and frequency stay.

Editing a QSO. A click on a row of the log table loads the QSO into the mask; the button is then called *Save*, with *Cancel* next to it. A double-click on the row opens QRZ.com instead.

Deleting a QSO. Via the action field of the table row, with a confirmation.

> ⚠️ **Caution:** A deleted QSO cannot be restored in LogRunner. If it was already uploaded to Wavelog, it stays there – deleting only works locally.

Search patterns instead of callsigns. In the callsign field `?` stands for exactly one and `*` for any number of characters (`??1EGL` finds every six-character callsign ending in `1EGL`). As long as a wildcard is in the field, LogRunner refuses to log and says so: "*Not logged – the callsign field holds a search pattern*". Without a wildcard, what you type counts as the beginning of a callsign.

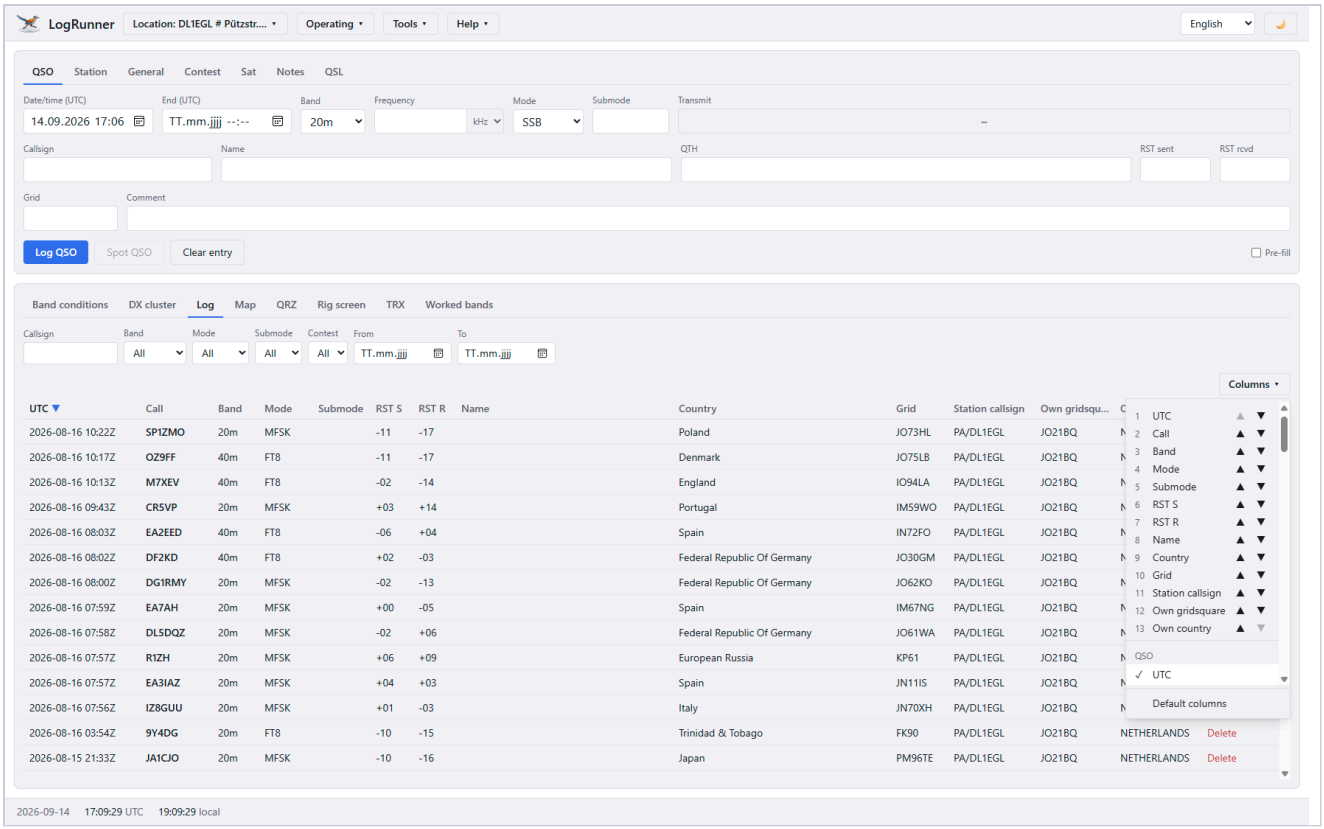

Fig. 7: The log table with filter row and column selection

4.2 Workflow 2 – Data exchange, import, export, interfaces

ADIF import and export

Tools ▸ ADIF import/export

Import: `Import ADIF …` opens the file dialog (`.adi`, `.adif`). Beforehand choose what should happen with duplicates:

Rule	Effect
Skip	Existing QSOs stay untouched
Update	The existing record is overwritten with the values from the file
Add anyway	The record is created a second time

Upper and lower case do not matter for callsigns. `DL1EGL`, `Dl1egl`, `dl1egl` and `dl1eGl` are the same callsign; the import recognises them as the same station and writes them into the log in capitals – the other station's callsign as well as *station callsign* and *operator*. The field *QSL via* stays untouched: in practice it holds free text (*via bureau*, *eQSL*, an e-mail address), which capitalising would spoil.

The report then lists: imported, updated, skipped, faulty records and the path of the automatically created backup. Fields LogRunner does not know are not lost – they are kept in `qso_extra` and written out again on export. Imported QSOs are assigned to the active location and put into the Wavelog queue.

> ⚠️ **Caution:** Before **every** import a complete backup of the database is created automatically in `~/LogRunner/backups/`. An import with the rule *Update* overwrites existing QSOs – the only way back is through that backup.

Export: `Export ADIF` writes to `~/LogRunner/exports/` into a file `logrunner-<YYYYMMDD-HHMMSS>.adi`. With the tick *Export only the filtered QSOs* the filter of the log table applies – callsign, band, mode, submode, contest and time range.

Wavelog/Cloudlog sync

Tools ▸ *Wavelog sync*

1. Enter the **Wavelog address** – just the logbook address, without `/api/…`; an endpoint path copied along is removed on save.
2. Enter the **API key**. The API generation follows from the key itself: tokens with the prefix `wl2_` use **API v2**, anything else **API v1**. For v2 the token needs at least the scopes `qso:read` and `qso:write`.
3. **Test connection** – reports the detected Wavelog version and, for v2, missing scopes.
4. Choose the **Station** (the Wavelog location number).

Button	Effect
Upload now	Sends all waiting QSOs
Fetch now	Fetches new QSOs from Wavelog, incrementally per location
Send changes	Overwrites QSOs in Wavelog that were changed here after uploading (API v2 only)

When something is pending, a table below the buttons lists the **waiting QSOs** – time, callsign, band, frequency, mode and both reports. The column *Kind* says which way each one takes: *New* goes out with *Upload now*, *Changed* with *Send changes*. A ⚠ in front means the last attempt failed; the QSO stays in the queue and goes along again next time. The table is for looking only – nothing is ticked or selected in it. With very many waiting QSOs it shows the first 200 and says below how many are not listed.

Without a connection the QSOs stay in the queue and go out on the next attempt and at the next program start. The number waiting is shown in the header.

Under *Station locations* each local location can be bound to a Wavelog logbook of its own; the button *Fetch from Wavelog* takes over the account's locations – as a preview that shows field by field what it would change.

> ⚠️ **Caution:** The API key is stored in plain text in `config.toml`. If the data folder lives on a USB stick or in a cloud folder, anyone with access to it can read the key.

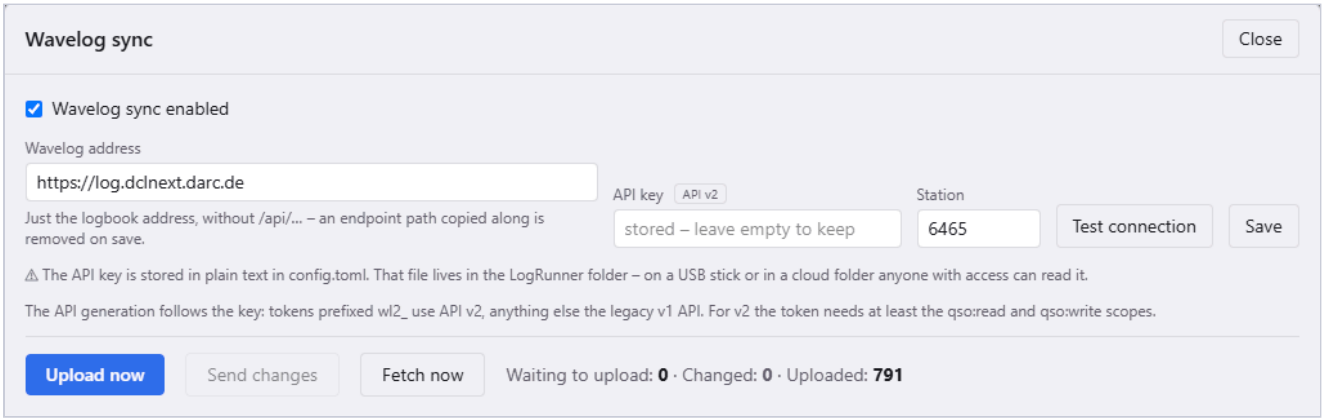

Fig. 8: The Wavelog sync with address, key and queue

Cabrillo submission

Tools ▸ *Contests* ▸ *Cabrillo submission* – see section 4.3.

DX cluster (Telnet)

Tools ▸ *DX cluster* switches the cluster on and manages the nodes. Three are preset (DXFun, VE7CC, NC7J); all three are perfectly ordinary, editable rows. Each node has an address, a port, a login callsign of its own and a list of commands sent after login.

The *DX cluster* tab offers three views:

- **Table** – the spots in order, with raw output and command line.
- **Band map** – a vertical scale per band, one row per frequency, with the VFO marker of your own rig and colours from the IARU band plan (CW blue, digital green, phone red, beacons yellow).
- **Band activity** – how many spots arrived per quarter of an hour, split by continent.

A click on a spot – in the list, in the band map, on the map or in the waterfall – takes callsign, frequency, band and, if given, the mode into the entry mask; date and time stay untouched. A **double-click** additionally opens QRZ.com. If TRX control is running with *Send a clicked DX spot to the transceiver* switched on, the same click tunes the rig.

Filter. Callsign, kHz from/to, band, mode, spotter, region, spotter region and comment. Wildcards: `*` for any number, `?` for exactly one character. Without a wildcard, callsign, spotter, region, band and mode count as the beginning, the comment as any part. Several values in one field are separated by a comma or semicolon (`FT8,FT4`) – except in the *kHz* field, where the comma is a decimal separator. Filter sets can be saved under a name.

Alarms. Rules of the form *field* + *pattern* signal visibly (counter on the tab) and audibly. For the frequency enter a range in kHz (`14000-14350`, `14000-`, `-7100`). Each rule can have a sound file of its own; a master switch and a volume control apply to all. Every **newly** arriving spot is checked, even when the window is showing something else.

Spotting yourself. *Spot QSO* in the entry mask pre-fills the spot form in the cluster panel.

> ⚠️ **Caution:** A spot goes out worldwide under your own callsign and cannot be recalled. Before sending, the form shows what the line will look like, and warns about self-spots and about a mode the band plan does not provide at that spot.

Skimmer spots (RBN) are off out of the box. A node with an RBN feed sends thousands per hour and would fill the buffer (500 spots by default) with machine spots within minutes.

Fig. 9: The DX cluster band map: one column per band, the spots at their frequency, colours from the band plan

WSJT-X link

Tools ▸ *WSJT-X link*

When switched on, LogRunner opens a UDP port and enters every contact logged in WSJT-X into the log at once – under the active station location, with band, mode, reports, locator and times.

In WSJT-X: *File* ▸ *Settings* ▸ *Reporting*, set "UDP Server" to the same address and port. The default is `127.0.0.1:2237`. If JTAlert or GridTracker runs alongside, only one of the programs ever listens – in that case enter the same **multicast address** in both programs, for example `224.0.0.1`.

> ⚠️ **Caution:** This is the only function that writes to the log without asking. That is why it is off out of the box, and why its state is shown in the status bar while it runs.

TRX control

Tools ▸ *TRX control*

Three connections are available – only one at a time, because the transceiver has one control port:

Connection	Used for	Settings
Icom CI-V (direct)	Icom rigs on the serial interface	Rig, port, baud rate, CI-V address (two hex digits, IC-7300: `94`, IC-705: `A4`)
Hamlib (rigctld)	All rigs Hamlib knows – including Yaesu, Kenwood, Elecraft, Flex	Host and port of the running `rigctld` (default `127.0.0.1:4532`)
Omnirig	When a second program needs the same transceiver	Rig1 or Rig2

Two switches decide what flows:

- **Take changes on the rig into the mask** – turn the VFO and frequency, band, mode and submode follow. A field the cursor is in stays untouched, as does a QSO opened for correction.
- **Send a clicked DX spot to the transceiver** – the only way in which LogRunner changes the rig's settings.

> ⚠️ **Caution:** It never controls the other way round. What is typed into the entry mask stays in the entry mask. A frequency typed by hand passes through 1, 14, 140 and 1400 on its way to 14.074 – a rig that followed along would sit in the broadcast band half the time.

Transmit state. If *Query transmit state (command 1Ch)* is switched on, the mask and the status bar show a red **On Air** while transmitting. Rigs older than about 2005 do not know the command; then the query should stay off.

Waterfall. *Fetch waterfall spectrum* shows the rig's spectrum in the **TRX** tab – 475 points per sweep, three to twenty sweeps per second depending on the baud rate (see below), with curve, frequency axis and VFO marker. A click into it tunes the rig. If the DX cluster is running, its spots are shown in the picture too – those that fall within the displayed span.

> ⚠️ **Caution:** The waterfall only works with Icom rigs with a built-in band scope and only through their USB or LAN port – on the `[REMOTE]` jack the rig rejects the command. Switching it on also switches on the band scope on the rig itself.
>
> ⚠️ **Caution:** The waterfall only shows the span the band scope is currently displaying – in **CENTER** that is the VFO frequency ± *span*, in **FIX** the edge range set on the rig. Spots outside this span are not in the picture and therefore cannot be clicked there either. With *span ±5 kHz* that is almost all of them: the span is then 10 kHz wide, not a band. To use the DX spots in the waterfall, set the span wide enough – or FIX with suitable edges.

Polling interval. How often LogRunner asks the rig for frequency, mode and transmit state. The default is 300 milliseconds; that is fast enough for the display and leaves the line some air. Shorter values bring little – the user

interface itself only looks twice a second anyway – but cost noticeable transmission time, which the waterfall then lacks.

Baud rate and waterfall share one line. One sweep of the spectrum is about 585 bytes. At 19200 baud that takes about 0.3 seconds, so at most a good three sweeps per second; at 115200 baud about twenty are possible. To get a smoother waterfall, raise the baud rate first – in the rig menu **and** in LogRunner; both must match. A polling interval of 100 ms, on the other hand, already uses about a fifth of the line at 19200 baud without making the display any faster.

Via the Icom Remote Utility. If the connection runs not over a cable but over the virtual COM port of the **Icom Remote Utility**, everything above applies – with one difference: the limit is then no longer the configured baud rate but what the network path between the two programs carries. A high baud rate there mainly means that the rig sends *more* data, and all of it has to go through that path.

Rig controls. Below the waterfall sits a folded bar with the transceiver's controls. Unfolded it has seven levels of sliders and ticks, switched like the tabs above: *Audio* (volume, squelch), *Receive* (RF gain, preamp, attenuator, AGC), *Filter* (noise blanker, noise reduction with level, auto notch, notch and notch position, filter, inner and outer PBT as centred sliders), *Monitor* (monitor and monitor level), *Transmit* (power, tuner, mic gain, compressor, VOX), *CW* (break-in, keyer speed) and *VFO* (VFO A, VFO B, A = B, A ⇄ B, split, RIT, XIT with **offset**, dial lock). The offset is a slider from −9990 through 0 to +9990 Hz with an input field next to it in which the value can also be typed. *VFO A* and *VFO B* latch and show which was chosen last – which one is active on the rig is not reported by the transceiver over CI-V. RIT and XIT share it, as on the rig.

> **Note on *Split*:** If the rig is set to a **repeater offset** (DUP− or DUP+ – on 2 m and 70 cm the auto repeater function sets it by itself), the *Split* tick stays empty, and that is correct: CI-V holds split and repeater offset in the same command, and a repeater offset is not a split. Up to 0.9.3b56 LogRunner read it as split – the tick then set itself again a few seconds after being cleared.

In addition there are two levels of buttons: **Mode** switches the mode (LSB, LSB-D, USB, USB-D · AM, FM · CW, RTTY), **Band** jumps to a band – to the spot that suits the current mode, because CI-V knows no band change, only frequencies. In both rows the button that currently applies is highlighted.

The band buttons are the ones the rig reports itself. When the bar is unfolded the rig is asked: first for its fixed band edges, and if it does not provide them, for its band stacking register. An IC-7300MK2 thus names twelve bands from 160 m to 4 m – 60 m included, for which it does not even have a band key of its own – an IC-9700 names 2 m, 70 cm and 23 cm. The edges come from the rig's regional version, not from a list in the program.

If a rig says nothing about this – older models, Hamlib, Omnirig – the full list is shown as before: 2200 m to 4 m plus **GEN**. And to see it on a rig that does answer, tick **Show all bands** under *Tools ▸ TRX control* – meant for a modified rig or a special permit the band edges know nothing about. **GEN** is only added if the rig can do shortwave at all; the button sets 10.000 MHz, and on a pure VHF/UHF rig that would just be an error message.

On the VFO level a **tuning knob** sits on the right that can be turned with the mouse (step 10 Hz, 100 Hz or 1 kHz); below it a frequency can also be typed and taken over with Enter – with a decimal point it is megahertz, without one kilohertz.

The **Segments / Bar** button at the top right switches the meters between a chain of boxes – like the display on the rig – and a continuous bar; the choice survives a restart. On every level the meters stand one above the other on the far right and stay in the same place when switching; on *Transmit* the column shows PO, SWR, ALC, COMP, Vd and Id, otherwise the S meter as well. The controls themselves are arranged in columns too – what belongs together stands one above the other. The bar is only available with **Icom CI-V**, not through Hamlib or Omnirig.

Meters. Above that stands what the rig is measuring right now: S meter, PO, SWR, ALC, COMP, Vd and Id – as bars, and where a characteristic curve exists, with the interpreted figure next to it. All seven are always shown; what is not being measured at the moment is pale and carries a dash instead of a number. While receiving only the S meter is

fetched, while transmitting only the transmit values – everything else would be zero anyway and would only cost the line the waterfall needs. Measuring only happens while the bar is open.

Sliders can be operated with the mouse wheel: one notch is one percent, five with Shift held. The value goes to the rig continuously, not only on release – throttled to at most five commands per second so the line does not clog up, and the last value set always arrives.

> ℹ️ ALC, COMP, Vd and Id are shown as a percentage of full scale. The rig measures Vd in volts, Id in amperes and COMP in decibels – the tooltip says so too – but the conversion differs from model to model, and a guessed number would be worse than an honest bar.

Three things about this:

- **It reads on unfolding, afterwards the bar follows the rig.** Turning a knob on the rig itself shows up here within half a minute – one value is re-read per poll so the waterfall keeps its line. The *Read again* button fetches everything at once.
- **Re-reading rests while transmitting.** Then the queries belong to the meters. On a sluggish connection too – via the Remote Utility, for example – the bar holds back so that frequency and transmit state keep priority.
- **What the rig does not know does not appear.** Each element is asked once; older rigs therefore show a shorter bar.
- **None of it transmits.** The tuner can be switched on and off, but not started – tuning stays the key on the rig.

> ⚠️ **Caution:** The combination of a high baud rate, the waterfall switched on and a short polling interval can overload the virtual port. It does not look like an overload: the Remote Utility still shows *connected* while the frequency display in LogRunner freezes. In case of problems, first set the polling interval to 300 ms, then switch off the waterfall and switch both back on one at a time.

Figure to follow: The waterfall with spectrum curve, frequency axis and DX spots shown in it

Callsign lookup

Tools ▸ Callsign lookup

Three services, all off out of the box:

Service	Account	Coverage
QRZ.com	Account required; the XML interface only serves the full record to accounts with the *XML Logbook Data* subscription	worldwide
HamQTH.com	free account required	worldwide, also delivers the DOK
callook.info	no account	US callsigns only (K, N, W, AA–AL)

They are asked in this order; each service is asked exactly once per callsign, and each field is filled by the first one that answers it. Answers are cached locally and are then available without a network too. The results land in the *QRZ* tab and fill name, QTH, grid, DOK and more into the mask.

> ℹ️ **For long-time LogRunner users:** the profile picture and the details next to it – website, licence class, LoTW/eQSL/QSL card – only exist since August 2026. Callsigns looked up before that are cached without them, and the tab therefore showed details without a picture – the picture only after pressing *Look up again*. The tab now does that by itself: it fetches such a record once the first time it is viewed. Meanwhile *looking up* is shown next to the callsign; the details are already there, the picture follows. On a client this does not happen – the cache does not live there.

In the *QRZ* tab text can be **selected with the mouse and copied with Ctrl+C** – not in the rest of the program, so that a drag across the band map does not leave a blue trail right through the panels. In the QRZ window with the complete page it works everywhere.

A **right-click** on selected text offers *Copy*; in an input field *Cut*, *Paste* and *Select all* are added. Where there is nothing to do, these four are left out. Ctrl+C, Ctrl+X and Ctrl+V do the same as before.

Set off by a separator, the **view** follows below – two or three entries with a tick at the one that applies, depending on what this window shows. Section 3.2 says what can be chosen there.

Again set off by a separator, there are always three switches – each *off* while the thing is shown, and *on* when it is gone:

Switch	Effect
Menu bar	The header with all menus (section 3.1)
Bottom status bar	The row with date, clocks and indicator lamps
Help texts	The short explanations under ticks and fields; tooltips at the mouse pointer stay in any case

They are available anywhere in the window and in every view, also over text and in an input field. That is why a right-click brings up a menu everywhere, even where none appeared before. Each of the three settings applies to *this* window only.

Below them is **Full screen on** – and while it is on, **Full screen off**: the window goes to the whole screen and back again. Up to 0.9.3b15 this was on **F11**; the function keys are meant to belong to the voice and CW keyer, so the key is free again.

Remember as startup state

The last entry, set off by a separator. It is the only one in this menu that records something permanently: everything above changes this window **now**, this entry decides what LogRunner opens with **next time**.

Until 02.09.2026 this happened by itself – clicking the close button at the top right recorded window size, window position and the chosen views. That had a price: anyone who just pushed their window aside once to look at something else, or who only wanted to try out a view, had changed the next start without meaning to. Now the rule is: **trying things out costs nothing; remembering happens on request.**

The entry writes in one go:

The window	Size, position on screen, maximised – plus the corner of the QRZ window, if one was open in this session
The view	Compact or Full; on a companion window the choice within its area
The window frame	Menu bar, bottom status bar, help texts
Language and colour scheme	the same two as in the menu bar
The map	all layers and the projection

The rest	Columns of the log table, band plan region of the band map, entry mode (log or contest), waterfall layout, band matrix by mode

Not included is anything that writes into a QSO or reaches into the network: the frequency unit of the input fields, *Pre-fill*, the station location, the QSO defaults and all settings under *Tools* – Wavelog, lookup services, DX cluster, LiveLink, CAT, screen capture. As before, those are written to `config.toml` immediately. The reason is a single one: a frequency unit that silently jumps back to MHz on restart because you forgot to remember it costs wrong QSOs – and no operating comfort is worth that.

The entry says what it did: where the menu was, **Startup state remembered** appears briefly. If the file could not be written, it says so in red.

On a Wayland desktop LogRunner takes a detour for this – since 0.9.3b43, and the tick for it is under *Tools ▸ Display ▸ Window places*. It is set out of the box: the window is then drawn through **XWayland**, the X11 layer that runs on every Wayland desktop anyway, and there it knows its place again. Any number of companion windows, each with its own mask, each remembering place and size – just as on Windows. Nothing needs to be set on the desktop for this.

The tick costs something: measured on a Raspberry Pi 5, three times per branch, about a tenth of the frame rate (49.6/50.0/50.8 against 44.0/45.6/45.2 fps under full load) and just under a tenth of a second at startup. Anyone who clears it – or whose computer offers no XWayland – gets the old behaviour: then remembering says *Saved as startup state – this display cannot remember the window place*, and size, view and everything else are recorded, only the place is not. The line under the tick says what currently applies.

Why this detour is needed at all: it is not a gap in LogRunner but the Wayland protocol. A window there does not learn its position and may not set it either – where a window goes is decided by the compositor alone. Measured on the Pi with the same test window under both displays: under X11, after moving it to 300/200 it reports exactly that, under Wayland it still reports 0/0.

Anyone who clears the tick and still wants their windows in a fixed place has two ways. Both were verified on the Raspberry Pi. With the tick set you need neither.

First: the compositor places the windows. With labwc this works with a window rule in `~/.config/labwc/rc.xml` – LogRunner identifies itself as `LogRunner`, and so does a companion window:

```
xml <windowRules> <windowRule identifier="LogRunner" title="LogRunner - Karte"> <action name="MoveTo" x="1920" y="0" /> </windowRule> </windowRules>
```

The task bar shows *LogRunner – Karte*: since b37 the companion window's name is part of the title, and each one can be addressed individually by it. The title carries **no version number** – otherwise the rule would point nowhere after the next beta. The main window is simply called *LogRunner*; a rule meant only for it needs `type="normal"`, otherwise it also hits the start window.

`ResizeTo` is not needed: LogRunner remembers the **size** itself under Wayland too, only not the place. Without `title` the rule applies to all LogRunner windows. After changing the file run `labwc --reconfigure` – that needs `LABWC_PID`, otherwise it aborts with *LABWC_PID not set*: `LABWC_PID=$(pgrep -x labwc) labwc --reconfigure`. The place is then fixed – moving the window with the mouse only lasts until the next start.

Where the numbers come from: from LogRunner itself. Start once via XWayland (below), place the windows, *Remember as startup state* in each – afterwards each `config.toml` holds the real place under `[window]`, and that goes into the rules. With the source package, use `tools/labwc_window_rules.py`: it reads all configurations and writes the rule block into `rc.xml`.

Second: start LogRunner via X11. In a Wayland session it then runs through XWayland, the X11 layer that runs anyway – and so the window knows its place again. *Remember as startup state* records it and the next start sets it, just as on Windows:

```
bash GDK_BACKEND=x11 ~/Programme/LogRunner/LogRunner
```

To make it permanent in the shortcut, put an `env` in front of the `Exec=` line:

```
Exec=env GDK_BACKEND=x11 DISPLAY=:0 /home/USER/Programme/LogRunner/LogRunner
```

`DISPLAY=:0` is required, otherwise LogRunner does not start. With `GDK_BACKEND=x11` GTK looks for an X11 display, and the desktop does not always pass a `DISPLAY` to a program started from the menu. Reproduced on the Pi: without it the start ends at once with *Gtk-WARNING: cannot open display* and exit code 1 – no window, no message. Which number is right is shown by `echo $DISPLAY` in a terminal of the running session; as a rule it is `:0`.

If nothing starts even so, the X11 key is missing. XWayland only lets in programs that can present a key from `~/.Xauthority`. `xauth list` must show a line for your own display there (`hostname/unix:0`). If it is missing, startup reports *Authorization required, but no authorization protocol specified*, and that too looks like *does nothing* from outside. Raspberry Pi OS recreates the file at every login (`/usr/bin/create_xauth`) – logging out and back in is therefore the simplest repair.

> **Companion windows do not reliably inherit the X11 branch.** Measured on the Pi: the hub ran via XWayland and remembered its place, the companion windows opened from it kept running under Wayland and remembered none. To take them along, write the same `env` prefix into their shortcuts – those are the files `LogRunner-<name>.desktop` in the program folder.
>
> Verified on the Pi: window moved to 420/260, *Remember as startup state*, and exactly that was in `config.toml` afterwards. On a screen without scaling there is no visible difference in the display; it is the same path LogRunner takes on every X11 desktop.

Like everything here, this applies per window: each companion window has its own `config.toml`, and the hub next to it keeps its startup state even when the client next door gets a new one.

Up to 0.9.3b4 the dialog *Tools ▸ Display* held the same three as ticks. They were dropped there: two ways to one setting are one too many, and the way through the Tools menu lies behind the menu bar it can hide.

With *Look up positions of cluster spots* the map places DX spots at their real location instead of the middle of the DXCC entity.

> ⚠️ **Caution:** Looking up passes the callsign you are working to a third-party server. The passwords of both services are stored in plain text in `config.toml` – so do not use a password here that unlocks anything else.

xOTA and SCP lists

Tools ▸ xOTA/SCP lists

The **Update** button fetches the current POTA, SOTA or IOTA list directly from the source; afterwards it is searched without a network – by reference or by name, directly in the fields *POTA reference* and *SOTA reference* under *General*. Taking one over also sets the reference's locator if the grid field is still empty. The lists are deliberately not shipped with the program – they change all the time.

Without a network it works by hand: download the file yourself and choose it, or put it into the data folder under the file name given.

The **Super Check Partial list** feeds the callsign suggestions in the mask. With the tick *Keep up to date automatically* it stays current by itself: at startup it is checked and fetched again as soon as it is a week old (about 360 KB each time).

LiveLink across several PCs

Tools ▸ LiveLink

> **Called *Netz-Synchronisation* up to version 0.9.3b8.** The name *LiveLink* is the same in all languages and is not translated. An existing setup is kept: the `[netsync]` section of an older `config.toml` is taken over at the first start.

One computer on the LAN runs the distributor as the **hub**, all others connect to it as **clients**. What happens on one PC goes to the hub and from there to all the others. The hub is found via mDNS; an address entered by hand beats the search.

The actual purpose is the **mask**: a computer shows exactly one area, full-screen – the map on the screen next door, the band map on the computer to the left.

Mask	Shows
(empty)	Everything – the whole window, and the way back

Mask	Shows
`cluster:table`	DX cluster · table
`cluster:map`	DX cluster · band map
`cluster:stats`	DX cluster · band activity
`logTable`	Log
`map`	Map
`solar`	Band conditions
`scope`	TRX
`workedBands`	Worked bands
`capture`	Rig screen
`qrz`	QRZ
`special`	Ticker

Band assignment

If several transceivers hang on several computers, it has to be settled **which one is meant**. The *band assignment* table sets this per band: one selection field per band, offering the connected PCs.

Three things depend on it:

- **A clicked DX spot** tunes the rig of the responsible PC – even if the click happened in a window on another computer, and even if the clicking window has no rig of its own.
- **The ^ key** (left of the 1) keys the transmitter of the PC responsible for the band in the entry mask – no matter which window has the focus. Reaching for the mouse to bring the waterfall window to the front first is no longer necessary.
- **Frequency and mode** of a client rig flow into the hub's entry mask when that client is responsible for the band. Without this, a QSO worked on the client rig and logged on the hub got the frequency of the hub rig in the log as soon as its knob was turned once.

The hub keeps the assignment and sends it to everyone; it can be changed on any computer. It is stored in `config.toml` under `[livelink] band_owner`.

> **Without an entry a spot follows the rig that is there.** If someone clicks a cluster spot in a window without a rig of its own, the window that has one tunes – whether hub or companion window. This has been so since 0.9.3b52 and was a gap before: if the transceiver hung on a companion window and the hub had none, nothing tuned at all.
>
> Anyone with **two** rigs on the network should therefore fill in the table. Without it both follow, which is rarely intended; with it exactly the responsible computer gets the command and all others stay out of it.
>
> ⚠️ If a rig transmits on request from the network and the connection breaks, it goes back to receive automatically. An "off" that no longer arrives would otherwise be a carrier that stays on.

For a **companion window on the same computer** the button *Create shortcut* creates a launch file. It starts a second LogRunner with its own data directory under `~/LogRunner/clients/<name>/` that shows only the chosen area.

> **Where the shortcuts live:** `~/LogRunner/shortcuts/`. Since 0.9.3b45, and that is a repair: before, they lay next to the program. A new bundle is unpacked next to it and the old one moved aside – the shortcuts disappeared from view with it, and after an update the companion windows were missing from the box *Start companion windows too*.
>
> At the first start of a new version LogRunner moves the files from the program directory there and rewrites them to the running bundle. The second is just as important as the first: a shortcut holds the **full path** of the program, and one that still points to the old one after an update silently starts the previous version. A renamed shortcut is found unchanged – it is copied, not rebuilt, but then keeps pointing to the old program.

Under *Connected PCs* each name shows whether it is the **hub** or a **client**. The hub cannot find that out by itself – its own window connects to it as a client like any other – so each computer states it itself when connecting. A PC running an older program version is shown as *client*.

Autostart of the companion windows

If this PC runs the hub, it can bring the companion windows up right away at program start. The box *Start companion windows too* lists every shortcut in the directory that *Create shortcut* writes to.

Presets. Which windows belong together depends on the use case – a contest evening wants different windows than portable operation. That is why the ticks belong to a **preset**: choose it at the top of the box, *New* (starts as a copy of the chosen one), *Rename*, *Delete*. Tick what belongs to this preset, and it is saved at once. At the bottom, **Autostart … with preset** decides which preset comes up at program start – regardless of which one is being edited at the top.

Start preset. The button on the right of the preset row brings the chosen preset up immediately – without restarting LogRunner. It also tidies up: running companion windows that do not belong to this preset are closed. Afterwards exactly this preset is on the screen. Going from contest evening to portable operation is one click instead of thirteen windows by hand.

No preset chosen	Then nothing starts, even if *Autostart* is ticked – and that is shown in red next to the selection. If the autostart preset is deleted, autostart stays without a preset; no other one is guessed
After the update	An autostart list from an earlier version becomes the preset *Standard* and is chosen for autostart – the same comes up as before
Saved immediately	Both ticks take effect on the spot – no *Apply* is needed, and a tick does not restart a hub
Hub only	The box only appears with the role *hub*. A client that started clients would send them to a distributor that does not exist on this computer
Once per program start	And only once the hub is really up. An *Apply* in the dialog does not start anything afterwards
Order	That of the list, i.e. alphabetical
Missing file	Stays in the list struck through and is marked *missing*. A renamed or moved shortcut should not disappear silently – otherwise a window would not come up one morning, and nothing would say why
Start preset	Brings up the preset chosen at the top – not the autostart preset: what you are looking at is what starts. Active while the hub is running. Below the button it then says what happened: how many windows were started, how many were already running and how many were closed
Closing the hub	When the window running the hub is closed, all companion windows of *this* computer go with it – whether they belong to a preset or were started by hand. Windows on other PCs stay open. This applies to an orderly exit via the close button; if the hub crashes, the companion windows stay up and reconnect as soon as it is back

What closes	Every running companion window of *this* computer that does not appear in the preset. A window that is in both presets stays open and is not restarted. Windows on other PCs on the network stay untouched – just like the transceiver, a window does not travel
A window that does not close	Is reported as such and not counted. The hub asks over the LiveLink connection; a companion window that is not connected at the moment does not hear the request
Renamed shortcut in the preset	Then nothing is closed at all, and the dialog says so. A renamed file does not reveal which window it starts – and without knowing that, none may be closed
Start	The button on the right of each row starts the window immediately – independent of the autostart tick, so also for a window that should not come along at program start. It is active while the hub is running. If the window is already running it is not opened a second time; *already running* is shown next to the button. A missing file has no button

> **Note:** The search goes by extension (`.lnk` on Windows, `.command` on macOS, `.desktop` on Linux) and not by name – a renamed shortcut is the same shortcut. `config.toml` holds only file names under `[livelink]` `autostart_shortcuts` ; an entry containing a path is left out when reading.
>
> ⚠️ **Caution:** The connection is unencrypted and asks for no password. It is meant for your own LAN, not for an open network. At the first start of a hub the Windows firewall asks for permission; on Linux the port may have to be opened by hand (`ufw allow 8765/tcp`), and mDNS needs the `avahi-daemon` there.

Rig screen via a capture card

Tools ▸ Screen capture

If the transceiver has an HDMI output with a capture card attached, the *Rig screen* tab shows its screen – complete, with menus and filter display. **ffmpeg** is required; a button in the panel fetches it by itself on Windows.

Setting	Meaning
Camera	The capture card. *Search again* re-reads the list.
Mode	Takes format, frame size and frame rate into the fields below
Pass through	Pass the picture on unchanged: best quality, hardly any CPU time, 10–50 Mbit/s at 1080p/30 depending on picture content
Re-encode	Smaller for Wi-Fi, at the cost of CPU time. The big levers are width and frame rate, not the quality level.
Offer the picture on the network	Off means: the picture server only listens on `127.0.0.1`
Access with key only	The key is part of the address and is generated when switched on

> ⚠️ **Caution:** *Run at program start* is deliberately off – addressing some cards wakes up the transceiver's screen.

Ticker – a screen that only shows things

Tools ▸ Ticker

Things that are often needed during operation and rarely operated:

Display	What it shows
Station clock	UTC on the left, local time on the right, each with seconds; below them the UTC date with the day of the year and, for local time, the zone abbreviation with its offset. Next to it rise and set of **sun and moon** for your own locator, in local time
Callsign and locator	Both from the **active station location** – switching the location also switches what is shown here
QSOs in the log	The total and the number of the past 60 minutes
Weather at the location	Temperature, humidity, air pressure, wind (speed at 10 m height and direction) and the probability of rain – each with the trend of the last three hours below
Alarm from the DX cluster	Callsign, frequency, band and mode of a spot that triggered an alarm rule – together with the rule itself

They are ticked individually. If several are chosen, they take turns **in the same window**; how long each stays is set by the selection field next to it – 10 to 60 seconds, the same for all. Dots below the tile show which of how many is currently visible. With a single display nothing changes, and the selection field is greyed out.

The clock keeps UTC, and not computed from local time: it reads it directly, so the display is right however the computer is set up. It ticks on the second boundary rather than at one-second intervals – so it does not fall behind after standby or switching windows – and catches up immediately when you return to the screen.

Sun and moon appear next to the clock as soon as a locator is entered in the active station location – calculated for its centre, shown in local time. If the sun does not set or does not rise at all on a day, that is shown instead of a time; north of the Arctic Circle that is the correct answer. For the moon the rise is missing on about one day a month – it rises later each day, and then none falls into the same day.

The weather comes from Open-Meteo and is fetched at most every 15 minutes. **It is only requested while the tick is set** – without it LogRunner opens no connection for it. The source is shown below the tile; Open-Meteo publishes its data under CC BY 4.0, and the attribution is the condition for that. If an answer fails to arrive, the last state stays and is marked as old after three quarters of an hour.

Under each value its line. It shows the last three hours, and LogRunner keeps it itself: every fetch is stored in a small file of its own next to the log (`data/weather.sqlite`, one week deep), because Open-Meteo does not provide its history in a way a display could pick it up along the way. For air pressure the line is the real use of the number: 1008 hPa says nothing on its own, 1008 hPa after 1016 says a front is on its way.

After a fresh start the line is still short – next to the source it then says how far back it really reaches, so nobody reads a temperature drop out of two points. Wind direction is the only one without a line: it runs in a circle, and a line from 350° to 10° would draw a turn around the whole dial that never happened. Deleting the file costs the lines and nothing else.

The alarm has priority. It does not take its turn but interrupts: as soon as a spot arrives that triggers one of your alarm rules from *Tools ▸ DX cluster*, it appears in the ticker – with a red border, for the display duration set. Then the rotation continues where it was. A new alarm resets the time, so the latest one is always visible.

For this the ticker fetches the spots itself while the tick is set – even on a window that does not show the DX cluster at all. Without the tick nothing interrupts, and without the cluster switched on there is nothing to interrupt.

On a client this works the same way. It has no connection of its own to a node – it gets the spots from the hub, and **the hub's alarm rules apply**. They are listed for reference in the alarm area of the cluster panel: what triggers is set up on the main computer and not on each screen separately.

The alarm is only audible on the hub. A client shows it – as a number on the tab and as a tile in the ticker – but it does not ring. One operator, several screens: a sound coming from two rooms says no more than one.

The filter, on the other hand, stays each screen's own. The hub provides what there is; what a screen shows of it, it decides itself – the wall screen in the next room may be set to 20 m while everything runs at the rig.

That is why the fields in *Tools ▸ DX cluster* are greyed out on a client: nodes, alarm rules and sound come from the hub, and a field that can be operated and does nothing is the kind of display that makes you look for the fault in the rig. **The dialog can still be opened, though** – looking is something other than changing, and anyone who wants to know which node the hub is connected to or which rules apply finds it there.

If a client should run its own cluster, the tick **Follow the hub** sits at the very top of the dialog. It is the one exception to the lock and set out of the box. Clear it and this window works independently **in the DX cluster**: own nodes, own filter, own alarm rules, own sound – the dialog opens up again for that. Everything else it keeps mirroring, log, map and entry mask included; the tick only concerns the DX cluster.

> Switching clears the spot list – from then on it has a different source, and a table does not show that it holds two times. And without its own cluster switched on, the *DX cluster* tab disappears: then there are no more spots, neither its own nor the hub's.

A screen of its own. The point of the whole thing is a second window that shows nothing else: a monitor above the rig with the clock on it. For this the LiveLink dialog offers the mask *Ticker* under *Shortcut for a companion window* – or on the command line `--client Uhr --mask special`. A **client** can show it too: it then gets callsign and QSO counts from the hub, because a client keeps no log of its own.

> Each window has its own selection. A companion window has its own data directory – so the screen on the wall may show the clock while the main window has the QSO counter up.

4.3 Workflow 3 – Contest operation

Starting a contest. In the *Contest* tab of the entry mask (or under *Tools ▸ Contests*) create a new contest:

Field	Meaning
Contest	The ADIF ID, for example `CQ-WW-SSB`
Exchange	The fixed part of your own exchange (zone, district, name …)
Counts a serial number	Off for contests that have none (CQ WW exchanges report and zone)
Number per band	Separate counting per band
Dupe counts	*per band and mode* (default), *per band* or *once at all*

Operation. *Operating ▸ ▸ Contest* switches the mask to the **contest line**. It shows only what is passed on the air, in exactly this order: Callsign · RST sent · Ser. sent · RST rcvd · Ser. rcvd · Exchange rcvd. The Tab key thus follows your ear, Enter logs from any field, and the cursor then jumps back to the callsign by itself. **Esc** discards the entry and does the same.

Your own serial number is counted, not typed: it is `max(stx)+1` over the session and is assigned when logging. It can be entered by hand in the *Contest* tab.

Dupes. If the station is already in the log for this contest, a red badge "*Already worked – hh:mm UTC on xx*" appears, and the QSO is **not** logged. **Ctrl + Alt + Enter** logs it anyway – the same key combination N1MM Logger+ uses for it.

> ⚠️ **Caution:** Whether a duplicate contact may count is the rule of the contest, not of this program. That is why there is a way past it – it is just not the one that happens by accident.

The *Full mask* button brings back name, QTH, grid and comment without ending the contest – for the one QSO in between that is not a contest QSO.

After the contest. *Tools* ▸ *Contests* lists all contests run. From there:

- **Resume** – the second morning of a weekend; numbers and dupes continue where they stopped.
- **Delete** – removes the session only. The QSOs stay in the log.
- **Cabrillo submission** – fill in the header data (callsign, categories, claimed score, operators, club, address, soapbox), check the *Preview*, *Write file*.

> ⚠️ **Caution:** The Cabrillo header is your declaration – none of it can be derived from the log, and a submission robot decides on it. Which form of exchange a particular contest expects is its own rule: check the QSO lines against the rules before sending.

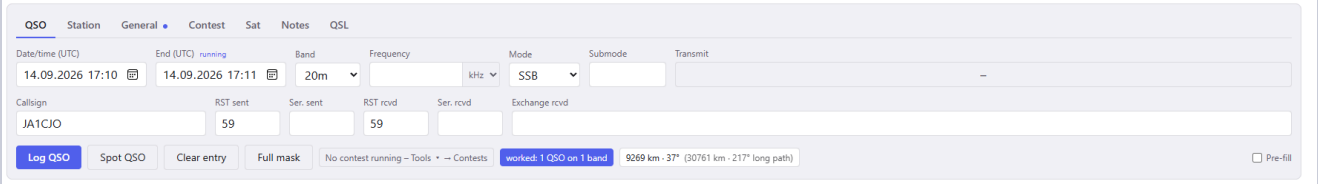

Fig. 10: The contest line with callsign, both reports and the exchange fields

4.4 Workflow 4 – Checking what has been worked

Worked bands. The tab of that name shows a band matrix for the callsign in the mask with the states *Worked*, *Confirmed* and *Not yet*. The tick *By mode* additionally breaks it down into CW, phone and digital. A click on a band cell opens the corresponding QSO list; from there a QSO can be opened directly for editing.

Log table. Above the table is the filter row: callsign, band, mode, submode, contest, from and to. Every column can be sorted by clicking its heading, a second click reverses the direction. The *Columns* button chooses what the table shows and orders the columns; at least one must remain.

Numbers follow the chosen language. Frequency, distance, transmit power and the antenna angles follow the language selection in the header – in English `14.074` (MHz) and `14,074,000` (Hz), in German the other way round. DXCC number, zones and the serial numbers of a contest stay without separators: those are identifiers, not quantities. **The input field of the mask does not follow this** – the frequency there always has a point, because a browser number field accepts nothing else.

The scroll bar covers the whole log – since 0.9.3b38; before it was the last 500 QSOs, which nobody noticed with a large log except when trying to scroll to the end. Rows are loaded while you scroll; where none is there yet, a grey bar stands in for it instead of wrong data. Sorting and filtering run over the **entire** stock, not over what is currently visible. With very large logs (beyond a hundred thousand QSOs) a jump into the middle of a table sorted by a rarely used column can take just under a second – the default *UTC descending* and the filter row answer at once.

Upper and lower case never matter for the callsign – since 0.9.3b38 also when looking up, no longer only in the filter row. A `db0jr` that sits in the log in lower case from an old stock is thus found by *worked before* and the band matrix just like `DB0JR`.

Statistics. *Tools* ▸ *Statistics* – total QSOs, callsigns, DXCC entities, continents and confirmed contacts as tiles, plus activity over 24 hours / 7 days / 30 days / 12 months, a band-versus-mode matrix, breakdowns by continent, year, location and operator, and IOTA progress.

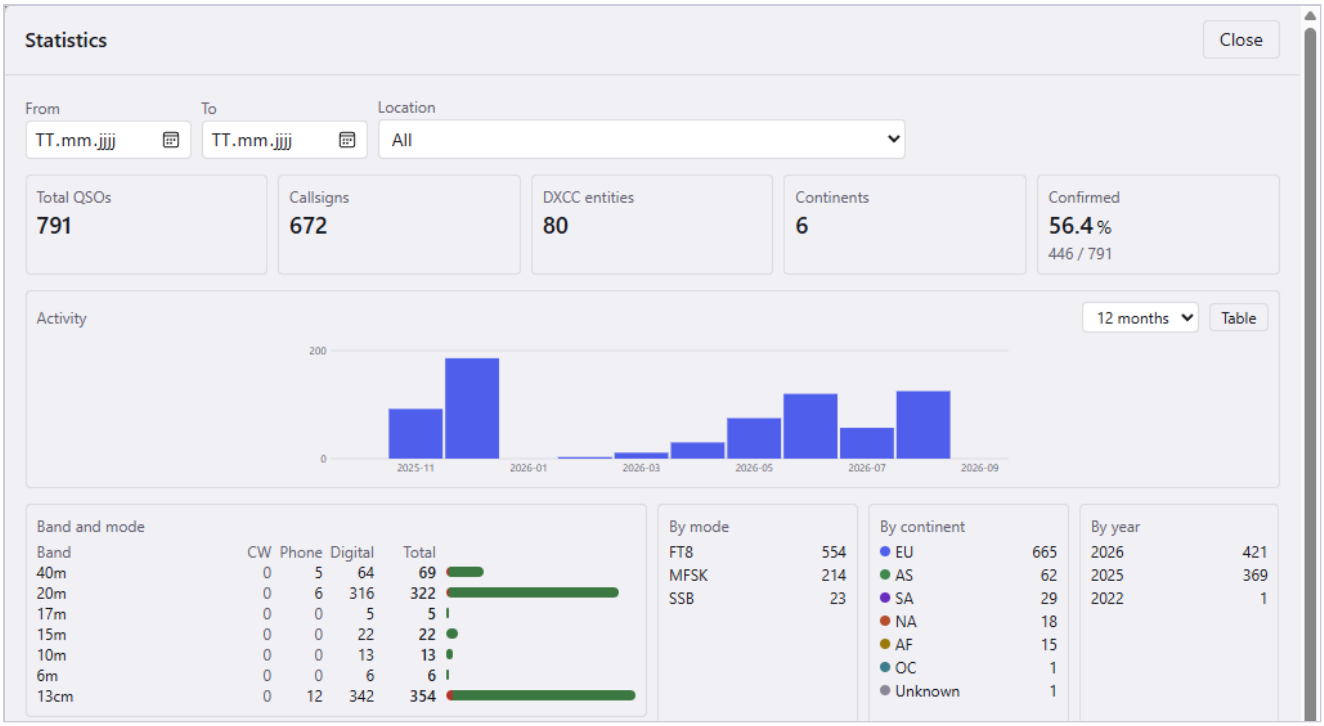

Fig. 11: The statistics with tiles, activity over time and band/mode matrix

One QRZ.com window for all windows on a computer. Opening a profile from a companion window shows it in the QRZ window of the main window – no second one opens. A client running on **another** PC keeps opening its own; the hub's window would be on a screen nobody is looking at.

4.5 Workflow 5 – The map

The *Map* tab shows where the other station is – roughly by prefix, precisely by locator –, your own station and the great-circle line between them. Coasts, islands, ice, lakes, reefs, rivers and borders come as vector data; no map service is queried.

Three views:

View	Special feature
Map	Flat world map, drag to pan, mouse wheel to zoom
Globe	The great-circle line is the arc it really is; drag to turn
Azimuth	The whole earth around your own station: every straight line through the centre is an antenna direction, every ring a distance, the edge the antipode at 20,015 km

Layers that can be shown: **Grey line** (twilight zone and night side for the current UTC time), **Prefixes** (DXCC entities, clickable), **CQ zones** and **ITU zones** (with borders and numbers, clickable), **Locator grid** (detail follows the zoom), **Places** (cities and towns by zoom level), **IOTA** (filled ring = worked, hollow = not yet), **MUF** (contour lines of the maximum usable frequency – see below), **DX spots** and **Log entries** (the QSOs the table is currently showing, with its filter).

A click on a spot or a log entry takes it into the entry mask.

> ⚠️ **Caution:** Without your own locator in the station location there is no distance, no bearing, no line – and the azimuthal map has no centre.

The MUF layer. The *MUF* button lays contour lines over the map: the highest frequency the ionosphere still reflects at that spot. Read in practice, the **14.0 line** says how far 20 m currently carries – whoever is inside it has the band. Six steps from 10.1 to 28 MHz, cool colours for the low bands, warm ones for the high bands.

Two things belong to it, otherwise the map is misread:

- **The number applies to a hop of 3000 km**, not to your path. It says *where* the ionosphere carries how much, not *whether* a particular contact will come about. Over two hops or over 500 km the result is different.
- **Age matters.** The data come from prop.kc2g.com, which recalculates every 15 minutes; the legend shows how old the data are. After one hour the lines are drawn **dashed and paler** and the age in red – a three-hour-old MUF contour is not roughly right, it is wrong.

Without a connection the last fetched state stays, with its age next to it; *Refresh* at the legend asks by hand.

> ℹ️ **Note:** The MUF layer reaches into the network and is therefore locked out of the box. When first switched on, the legend says *Fetching not enabled* and next to it is the button **Fetch data** – one click enables the source and fetches at once. After that the map button only switches the display. (In `config.toml`, `enabled = true` in the `[muf]` section does the same.)

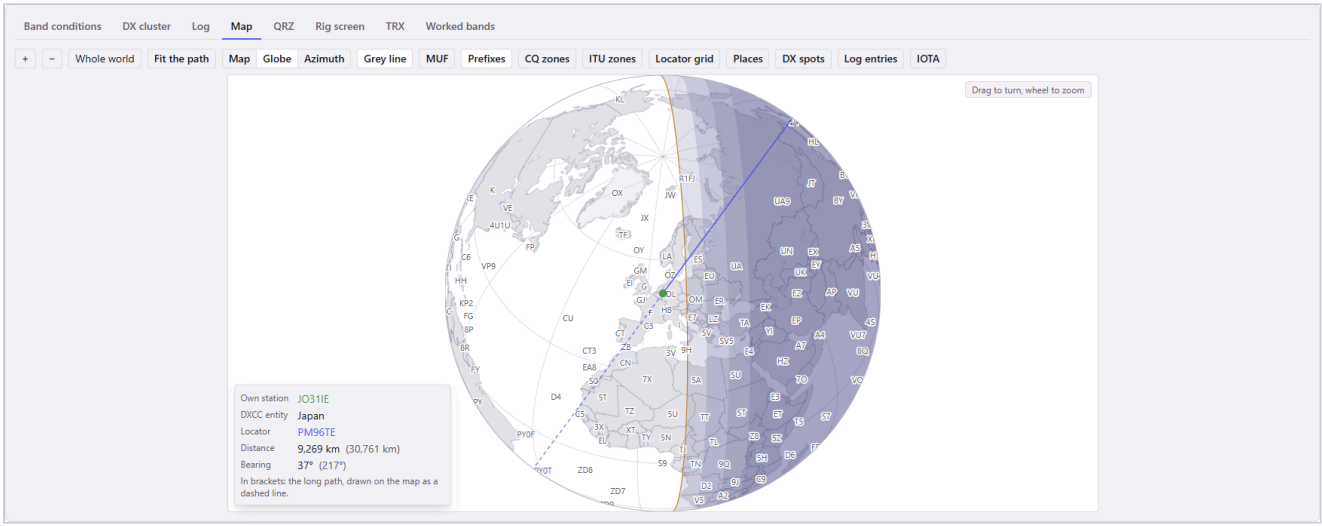

Fig. 12: The world map as a globe with great-circle line and prefixes shown

4.6 Workflow 6 – Band conditions

The *Band conditions* tab answers three different questions, and they appear in this order one below the other:

Area	The question behind it
The figures	What is the situation at all?
Path indicator	Which bands carry to **this** station?
Band indicator	Where can I get to at all right now?

The first comes from N0NBH and is a measurement, the other two are a calculation against the MUF contours from KC2G. That is not a detail but the difference between *how is the sun* and *does my signal arrive there*.

The figures

Fetched on request from hamqsl.com: solar flux, sunspots, A and K index, MUF, X-ray flux, state of the geomagnetic field and noise level, plus the rating per band group for day and night. It is queried at most hourly; the answer is stored locally and is available without a connection too – then with its timestamp and the note *outdated*.

LogRunner additionally stores one reading per hour and derives **trend arrows** and an overall trend (0–100) from them, meant for comparison with itself.

The MUF card may come from elsewhere. hamqsl.com has been delivering this one field empty for quite some time. If it stays empty and the map's MUF layer is enabled (section 4.5), LogRunner fetches the value from prop.kc2g.com – from the ionosonde nearest your locator. The card then carries **· KC2G** and names station, distance and age in the tooltip; below the panel is a second source line. Without your own locator, without a sonde within 3000 km or as long as the source is not enabled, the dash stays – each time with a note saying which of the three reasons it is. A number from far away would be no answer to "what works here right now", and a dash without a reason looks like a fault.

Path indicator

The same question a second time, but for **one** path. What it calculates for is set by the two fields next to it:

> Carries now [callsign or prefix] **/** [QTH locator]

The **first** takes a callsign or just a prefix – *VK* is enough, *VK3ABC* works too. If it stays empty, the callsign from the entry mask applies; that is the normal case when logging, and the field is there for the side question: *and what if I called VK right now?*

An entry in the callsign field of the QSO mask puts this side question back and brings the path indicator along to the new station: what carries to it right now is the most common question when logging, and it should be answered without a second step.

The **second** takes the other station's QTH locator, and it is the more precise information: a callsign names a country at best, and the middle of W or UA lies a thousand kilometres from the station. If a locator is given, the calculation runs against the place instead of the country – it beats the callsign.

If the locator is already known from the entry mask's callsign, it is shown in grey as a placeholder in the field. That means *this one applies right now*, and you do not have to clear it first to enter another – the same rule as with the callsign field next to it.

And if you type a callsign of your own into the path indicator, LogRunner looks up its locator itself. Four sources, in this order:

Order	Source
first	the **lookup cache**: if the callsign was already looked up at QRZ.com or HamQTH, its locator is there
then	**your own log**: the most recent QSO with this station that carries a locator
then	**QRZ.com or HamQTH**, freshly queried – if set up and the network is reachable
last	**cty.dat**: the middle of the DXCC entity, converted into a locator

The first two and the last answer from disk and at once. Only the third can take seconds, and it is therefore asked **only here** – not on every keystroke in the entry mask, and only for a **whole** callsign: a prefix has no entry with any of the services.

The last source looks different from the other three. A locator from the cache, the log or QRZ describes a station and is therefore shown as a value in the field. The middle of a country is not a station – it is only shown **in grey as a placeholder**, and the text under the mouse pointer says where it comes from. The calculation and drawing still use it; the crosshair ⌖ in the line next to it keeps saying *middle of the country, not the station*.

The same display applies to the callsign from the **entry mask**: if a station is there whose locator LogRunner knows, it appears here in grey as a placeholder – even if it is not in the mask itself, because it comes from the log or the

cache. The locator shown is always the one actually used in the calculation, and the text under the mouse pointer names its source.

This also applies when the path indicator currently **cannot rate anything** – because the MUF contours are missing or fetching is not enabled. Which place is meant does not depend on the ionosphere; an empty field there would mean *no locator known*, and that would be something else.

If no station locator is found, the field is cleared – even if one was in it a moment ago. That is not carelessness but the point: the locator beats the callsign, and an old locator next to a new callsign would win the calculation. *VK* would be displayed, *JO31* calculated.

If you know the locator, enter it by hand afterwards – LogRunner does not overwrite that.

Two subtleties about the cache and the log: the cache searches by the **base form** of the callsign (DL/W1AW/P is W1AW there, because the licence is the same), the log by **exactly what was typed** (DL/W1AW/P was somewhere other than W1AW, and the locator from back then belongs there). For a callsign with an attached prefix or suffix, however, the cache is **not asked for the place at all**: W1AW's record names Connecticut, and that is exactly where DL/W1AW/P is not. Then the country named by the attached prefix applies.

As long as what is in the locator field is not yet a complete locator – *JO3*, say – the field gets a red border, and what applied without it continues to apply. Two to eight characters make one.

The × clears both fields again – they ask **one** question, and whoever puts it back puts it back entirely.

Two lines on the map, and they have nothing to do with each other

The QSO path is the line to the station currently in the QSO entry mask: solid the short path, dashed the long one. It appears as soon as a callsign or part of one is there, and it stays as long as something is there – no matter how it got there: typed, by clicking a DX spot in the table or on the map, or by clicking a log entry.

The path indicator's line is the path asked about in the path indicator – dotted and in a colour of its own. It appears as soon as something is in one of the two fields or the map has been clicked, and **only with the MUF display switched on**: it is the path over which the indicator rates its bands, and without contours there is nothing to rate.

Neither takes anything away from the other. Anyone who wants to know in the middle of a QSO what currently carries to Japan types it into the path indicator or clicks on the map – the QSO path stays where it is. Until 02.09.2026 this was different: there was only one line, and LogRunner first asked whether the entry mask was still needed. **That question no longer exists.**

Below the map both paths are listed one above the other: at the top the QSO with country, locator, distance and direction, set off below it the path indicator's path.

With the MUF display switched on, the reflection points at the F2 layer sit on the path indicator's line – one per hop, in the same colour as the line. The limiting one, where the path closes, gets an amber ring. They lie where the indicator calculates: if it has no question of its own, that is the QSO path, otherwise its own line.

The click on the map

The most convenient way to a locator is a click on the map. As long as the MUF display is on, a click anywhere puts the locator of that spot into the path indicator's field and draws its line there. A callsign that was in the path indicator is deleted – it has nothing to do with the clicked point. The QSO path stays untouched.

As precise as the view allows: two characters on the whole world map, four zoomed in, six very close – the same level at which the map also draws its locator grid. Six characters from a click on the world map would be two guessed ones.

A drag across the map is not a click, and a clicked DX spot is not a map point: that is meant for itself and goes into the QSO entry mask as always.

Three ways lead away from a clicked point again: the × in the path indicator, an entry of your own in one of its two fields – and **an entry in the callsign field of the QSO entry mask**. Anyone entering a callsign there wants to know what carries to *this* station right now; the path indicator is then brought along to it, and its two fields are empty again. The QSO path on the map does not change – it follows the entry mask anyway.

Merely clearing the mask – after *Clear entry* or after logging – does not count: that is not a new target but none at all, and an open question in the path indicator then stays.

One box per band from 160 m to 6 m: a filled dot ● means open, a half ◐ marginal, an empty ○ closed. 8 m is left out on purpose – the band is not allocated to amateur radio in any of the three ITU regions and is in the log only because ADIF lists it for the few national experimental permits. The line above shows the number of hops, the distance, the path MUF and the age of the data; the mouse pointer on a box names the margin to the MUF and, for a closed band, the place on the path where it closes.

The difference from the table above is essential: that one comes from N0NBH and says what conditions are like in general. This block calculates against the MUF contours from KC2G along the great circle to the other station and says whether the ionosphere reflects the frequency **there**. It does not predict a contact: attenuation, noise, antennas and power are not included, and a band can be open and still unusable. Anyone needing a real path budget finds it in the HF Propagation Planner on prop.kc2g.com.

The display names two limitations itself. If more than half of the path lies in daylight, 160 m, 80 m and 40 m are shown dimmed instead of green – the D layer eats them up by day, even if the MUF would carry them. And above 10 m the method only calculates the F2 layer: on 6 m *closed* therefore means **no F2**, not *no opening*, because it knows nothing of sporadic E and aurora. An *open* there, conversely, is serious information.

Band indicator

Below it the same traffic light appears again as a grid: vertically the continents – alphabetically AF, AN, AS, EU, NA, OC, SA –, horizontally the bands, in the same boxes as in the path indicator above. This table is tied to **no** callsign; it answers the other question, namely *where can I get to at all right now*. An empty cell does not mean *closed* but *no information*.

A continent is not a place, and the display does not claim so either. The calculation runs against several points per continent – for Asia, for example, Japan, eastern China, India, Siberia and the Middle East –, and per band the **best** of them is shown. A ● therefore means: *somewhere there it carries*, not *everywhere*. Which point carries is shown by the mouse pointer, together with distance, direction and path MUF – enough to turn the antenna. The detour via several points is necessary because a continent spans many time zones: between Japan and Turkey lies half the ionosphere at the same hour, and a point in the middle answers neither of the two questions.

The calculation uses the same contours the map's MUF layer draws, and therefore everything said there applies here: fetching is locked out of the box, and without an enabled source the block stays empty and says so.

The points on the path line

With the MUF layer switched on, the map marks small points on the line to the other station – one per hop.

They lie where the ray is **reflected** at the F2 layer, i.e. in the *middle* of each hop – not where it touches the ground again. That is the spot where the state of the ionosphere decides about this hop, and exactly there the path indicator calculates.

It follows how to read them:

- The **number** of points is the number of hops.
- The **distance between two neighbouring** points is exactly one hop length. That is where it can be read off.

- The distance **from your own station to the first** point, on the other hand, is only **half** a hop length – and the same at the other end. Measuring from there gives half.

An example: 9106 km to Thailand are three hops of 3036 km each. The three points lie at 1518 km, 4553 km and 7589 km – 3036 km between each of them, 1518 km each towards the ends.

One of the points may be **amber and larger**: that is the narrowest spot of the path, the hop with the lowest MUF. It determines the whole path, because a hop that does not come back closes the way – the answer to "why is 20 m closed" is thus a *place*. If the whole path lies within the same contour step, there is no narrowest spot; then all points stay blue, and that is not a missing display but the correct information.

A window for a single part

On a companion window with `--mask solar` the right-click menu lets you choose which of the three parts is shown alone – *Solar activity*, *Path indicator*, *Band indicator* or *Detail* for everything together. The window then measures its height again. Section 3.2 lists the entries in context, section 4.2 the LiveLink behind it.

Where the figures come from

The technical foundation, in case a display does not say what it should.

Display	Source	Fetched	Stored in
Figures, band table, VHF	hamqsl.com (N0NBH), `solarxml.php`	at most hourly	`data/solar.json`
Trend arrows, overall trend	the same figures, recorded locally	one record per hour, 30 days	`data/propagation.sqlite`
MUF card (fallback source)	prop.kc2g.com, `api/stations.json`	with the contours	`data/muf.json`
Path and band indicator, the map's MUF layer	prop.kc2g.com, `mufd-normal-now.geojson` and `fof2-normal-now.geojson`	every 15 minutes	`data/muf.json`

Both fetches are **locked** out of the box and are enabled separately: `[solar] enabled` for the figures, `[muf] enabled` for the contours. Locked means: nothing is fetched, but an existing cache is still displayed. What was fetched once is yours.

How a contour becomes a traffic light, in five steps:

1. Location of the other station from the locator, otherwise from `cty.dat`; from there the great circle to your own station.
2. The path is divided into hops of at most **3500 km**; the control point is the middle of each hop.
3. For each control point the contours say between which two steps foF2 and MUF lie. The conversion to the actual hop length uses the secant law, with a fixed assumed reflection height of **300 km**.
4. The **path MUF is the minimum** over all control points – a hop that does not come back closes the whole path.
5. The traffic light: **open** up to 0.85 × MUF (the classic *frequency of optimum traffic*), **marginal** up to the MUF, above it **closed**. It is calculated for the bands from 1.8 to 54 MHz, without 8 m.

What the method cannot do – fundamentally, not out of carelessness:

- **No attenuation, hence no LUF.** There is no D-layer model; the daylight mark on 160/80/40 m is a rule of thumb, not a model. By the method alone, 80 m would always be open at noon.
- **No noise, no antennas, no power.** It says *the ionosphere carries*, not *the contact will come about*.

- **The resolution is that of the contour steps**, and they lie on the band frequencies. For the question "does 20 m carry" that is exactly right; an MUF to 0.1 MHz is not in there.
- **No forecast.** Everything refers to now. For 24 hours ahead there is the HF Propagation Planner on prop.kc2g.com.

Modules involved, in case you look into the source code: `solar.py` (feed and cache), `propagation.py` (the month of readings), `muf.py` (contours, station values, enabling), `pathmuf.py` (the calculation from the five steps above), plus `api/solar.py` and `api/muf.py` as the bridge to the user interface.

4.7 Workflow 7 – QSO defaults

Tools ▸ *QSO defaults* manages named sets of start values – for example "FT8 Digital" with mode, submode and reports. The active set pre-fills the empty mask; overwriting and deleting is possible there at any time. Fields of the other station are deliberately not included.

4.8 Workflow 8 – Adjusting the display

Tools ▸ *Display* (applies to this window only and survives a restart):

Setting	Effect
QRZ.com profiles	Open in a LogRunner window or in the system browser
Band plan	IARU region 1, 2 or 3 – divides the band map's columns

Menu bar, bottom status bar and help texts were here up to 0.9.3b4 and now depend on the **right-click** alone – see *Callsign lookup* in section 4.2.

4.9 Workflow 9 – Delete all QSOs

Tools ▸ *Delete all QSOs*

> ⚠️ **Caution:** This deletes the **entire** log. All QSOs with their extra fields and sync status are removed from the database. It cannot be undone in the app.

What happens:

- Beforehand a complete backup of the database is created automatically in `~/LogRunner/backups/`. **If this backup cannot be written, nothing is deleted.**
- Kept: station locations, Wavelog credentials, language and theme.
- The Wavelog pull cursor is reset so *Fetch now* can bring the QSOs back from there if needed.
- To confirm, type the exact number of QSOs present. If it does not match, nothing is deleted.

Restoring: close LogRunner, copy the desired file from `~/LogRunner/backups/` back to `~/LogRunner/logrunner.sqlite`, start LogRunner again.

5. Settings & Configuration

5.1 Where the settings live

All settings are stored in `~/LogRunner/config.toml`. The file is created with the defaults at first start and maintained by the program afterwards; every change in a dialog ends up there. It can also be edited by hand – then with the program closed.

Every **companion window** has its own `config.toml` under `~/LogRunner/clients/<name>/`. That way each window remembers its own size, its own corner and its own area.

> ⚠️ **Caution:** `config.toml` holds the Wavelog API key and the passwords for QRZ.com and HamQTH in **plain text**. There is no encryption in this version. Anyone who can read the file can read the credentials.

5.2 The sections of `config.toml`

Section	Content
`[general]`	Language, theme, view, entry mode, columns, map layers, band plan region and other window matters
`[window]`	Size, place on the desktop and whether the window was maximised *(written by the program)*
`[wavelog]`	Address, API key, location number, pull cursor per location
`[lookup]`	QRZ.com, HamQTH, callook.info, spot positions, timeout
`[wsjtx]`	On/off, address, port
`[cat]`	On/off, connection, polling interval, direction switches, waterfall, one subsection per connection
`[livelink]`	On/off, role, hub address, port, PC name, mDNS, mask, band assignment, autostart of companion windows
`[capture]`	On/off, device, format, output, network sharing, key, ffmpeg path, picture source
`[solar]`	On/off of the hourly fetch of the solar figures
`[muf]`	On/off of fetching the MUF contours from prop.kc2g.com
`[dxcluster]`	On/off, auto connect, login callsign, buffer, view, expanded area, three expiry times, skimmer, filter, filter sets, alarms, server list

5.3 Important individual options

Key	Values	Default	Meaning
`general.language`	`de`, `en`, `es`, `fr`, `it`, `pt`	`en`	Language of the user interface
`general.theme`	`dark`, `light`	`dark`	Colour theme
`general.view`	`detail`, `compact`	`detail`	How much of the window is shown
`general.entry_mode`	`log`, `contest`	`log`	What the entry mask is for
`general.menu_bar`	`true` / `false`	`true`	Menu bar at the top (header with the menus)
`general.status_bar`	`true` / `false`	`true`	Bottom status bar
`general.show_hints`	`true` / `false`	`true`	Explanatory texts under fields

Key	Values	Default	Meaning
`general.prefill`	`true` / `false`	`false`	Keep band/mode/submode/frequency
`general.freq_unit`	`Hz`, `kHz`, `MHz`, `GHz`	`MHz`	Display unit of the frequency fields (always stored as MHz)
`general.qrz_open_in`	`window`, `browser`	`window`	Where a QRZ profile opens
`general.bandmap_region`	`1`, `2`, `3`	`1`	IARU band plan for the band map
`general.scp_auto_update`	`true` / `false`	`true`	Keep the SCP list up to date automatically
`general.frontend_source`	`server`, `file`	`server`	*Set by the program* – see section 6
`wavelog.enabled`	`true` / `false`	`false`	Wavelog sync
`lookup.timeout_s`	number	`4.0`	Timeout of a callsign lookup
`wsjtx.host` / `.port`	address / number	`127.0.0.1` / `2237`	Where to listen for WSJT-X
`cat.backend`	`icom`, `rigctld`, `omnirig`	`icom`	Connection to the transceiver
`cat.poll_ms`	number	`300`	How often the rig is polled
`cat.qsy_on_spot`	`true` / `false`	`true`	Send a clicked spot to the rig
`cat.adopt_from_rig`	`true` / `false`	`true`	Rig → mask
`cat.all_bands`	`true` / `false`	`false`	Show all band buttons instead of the bands the rig reports
`livelink.port`	number	`8765`	Port of the hub
`livelink.autostart`	`true` / `false`	`false`	Bring up companion windows when the hub starts
`livelink.autostart_presets`	list of `{name, shortcuts}`	empty	The window presets; `shortcuts` are file names
`livelink.autostart_preset`	name	empty	Which preset autostart starts; empty = none
`livelink.autostart_shortcuts`	list of file names	empty	The windows of the autostart preset, written along for older versions
`capture.port`	number	`8770`	Port of the picture server
`dxcluster.max_spots`	50–5000	`500`	Spots in the buffer
`dxcluster.show_skimmer`	`true` / `false`	`false`	Accept RBN spots
`dxcluster.*_retention_minutes`	`0`, 5, 10, 15, 30, 60	`0` (never)	Expiry time for band map, map and waterfall – one each

5.4 Environment variable

Variable	Effect
`LOGRUNNER_HOME`	Sets the data folder. Without it `~/LogRunner` applies.

5.5 Command line

Unknown options are skipped and logged, not punished – a window program without a console could not write a usage note anywhere.

Option	Effect
`--client [name]`	Starts a **companion window** with its own data directory under `~/LogRunner/clients/<name>/`. Without a name the next free one is assigned.
`--mask <id>`	Which area this window shows alone (see the table in section 4.2)

Example:

```
LogRunner.exe --client Bandmap --mask cluster:map
```

5.6 Keyboard shortcuts

Key	Where	Effect
Esc	open tool dialog	Close the dialog, cursor back to the callsign field
Esc	contest line	Discard the entry, cursor back to the callsign
Esc	cluster spot form	Close the form without spotting
Esc	open menu	Close the menu
Enter	entry mask	Log QSO (in the contest line from any field)
Ctrl + Alt + Enter	contest line	Log a dupe anyway
^ (left of the 1)	without focus, in the callsign field, in the contest line also in *Ser. rcvd*	Transmit (PTT) – on your own rig, or via the band assignment on the rig of another PC
Space	contest line: callsign or *Ser. rcvd*	Jump back and forth between callsign and *Ser. rcvd*. If the contest counts no numbers, it jumps to the *Exchange rcvd* field instead
Tab	entry mask	To the next field; your own serial number is skipped
↓ / ↑	menu button	Open the menu and navigate in it
↑ / ↓	divider in the waterfall	Fine-tune the split between curve and waterfall
Double-click	divider in the waterfall	Reset the split
Double-click	callsign field, log row, DX spot, map	Open the QRZ.com profile
Single click	DX spot, log entry on the map	Take it into the entry mask
Mouse wheel	map	Zoom

> ⚠️ **Caution:** The ^ key switches the transmitter. It therefore works explicitly **only** when nothing has the focus, the cursor is in the callsign field or – in the contest line – in *Ser. rcvd*. In name, QTH and comment fields ^ stays a character. The key is recognised by its position left of the 1, not by the character on it: on a US keyboard it is the key with the `, on a French one the key with the ². Without connected TRX control nothing happens at all.
>
> **Up to 0.9.3b55 it was the space bar.** Since then, in the contest line, that belongs to the jump between callsign and *Ser. rcvd* – as in every contest logger.

F1 to F12 are deliberately unassigned. In every contest logger these keys carry the CW macros; LogRunner leaves them free so a keyer can take them over unchanged later.

6. Troubleshooting

6.1 Where the log files are

File	Content
`~/LogRunner/logs/logrunner.log`	The running log. Rotating, at most 512 KB per file, two backups. At the beginning of every start it contains an environment report (version, Windows, Python, packaged yes/no, WebView2 version, display engine).
`~/LogRunner/logs/start.log`	Start errors only – the cases in which no window ever appeared.
`%TEMP%\logrunner-start.log`	Emergency location, in case not even the data folder can be created.

Help ▸ *About* shows the same system information and copies it to the clipboard at the press of a button – this information belongs in every bug report.

More detailed logging for troubleshooting. Normally LogRunner only writes what matters. Anyone reproducing a fault who wants to see more starts once with the environment variable **`LOGRUNNER_DEBUG=1`** set – then the small stuff is in the log too:

```
$env:LOGRUNNER_DEBUG = 1
& "C:\LogRunner\LogRunner\LogRunner.exe"
```

On the Pi accordingly `LOGRUNNER_DEBUG=1 ~/Programme/LogRunner/LogRunner`. The log gets considerably longer; for everyday use the switch should go away again.

The user interface then writes along as well. If, for example, you click a cluster spot and the transceiver does not follow, the log contains a line `Oberflaeche [qsyToSpot]: …` – it names the path the click took (sent to another computer, no rig of its own, *QSY to spot frequency* switched off, or tuned here) and the values that led to it. That is the difference between a guess and a finding: without this line three of the four paths look the same from outside – nothing happens.

6.2 Startup problems

Message	Meaning and remedy
"The Microsoft Edge WebView2 Runtime is missing on this computer."	Without it pywebview would fall back to the old Internet Explorer engine, in which the user interface does not run. Install the *Evergreen Standalone Installer (x64)* for WebView2 and restart.
"The Microsoft Edge WebView2 Runtime … is version n. LogRunner needs at least version 111."	A runtime that exists but is too old. It is worse than none: everything starts, but the user interface stays a black rectangle. The same installer brings it up to date.
"LogRunner is already running (lock: …)"	An instance is already running on the same data directory. Two processes on the same SQLite file are the one way to destroy a log that no backup is fast enough for. Look for the running window – if necessary in Task Manager for `LogRunner.exe`. A lock file left over from a crash holds nobody up.
"Database damaged: … Please restore from a backup in …"	The integrity check of the SQLite file failed. Close LogRunner, copy the most recent file from `~/LogRunner/backups/` to `~/LogRunner/logrunner.sqlite`, restart.

Message	Meaning and remedy
"The settings file cannot be read: …"	`config.toml` was edited by hand and is now faulty. Correct the named spot – or rename the file; LogRunner then creates a new one with the defaults. **The credentials must then be entered again.**
"The web part of the program is missing: … is not there. Please unpack the archive completely."	The `_internal` folder is missing or incomplete. Unpack the ZIP archive again, completely.
The window stays empty or black	LogRunner detects this itself: it queries the page, waits up to 25 seconds and then automatically tries a second way – the files directly from the folder instead of through the local web server. If that works, `general.frontend_source = "file"` is recorded in `config.toml`, and the next start takes that way right away. If it does not, a message box appears with display engine, page state and reported errors. Most common causes: a WebView2 runtime that is too old, or a computer that does not let the program read its own files (virus scanner, group policy).
The start window asks for the licence at every start	The record `~/LogRunner/license.json` could not be written – a read-only data directory, a stick without write permission. The log contains a line about it. The program runs anyway; only the question comes back.
"The file LICENSE.txt could not be found."	It belongs next to `LogRunner.exe` and is missing there. Unpack the archive again completely. The question about consent is asked anyway – the agreement applies without the file too.
Two LogRunner windows show the same thing although one is newer	Up to 0.9.3b2 several windows on one PC shared a web server – the second loaded the user interface of the first. Since then each window gets its own port; the log says *Oberfläche wird auf Port N ausgeliefert* (user interface served on port N).
After a restart the QRZ login is gone again	The folder `~/LogRunner/data/webview/` could not be written; the display library then runs in private mode. The log contains a line about it. Nothing else is affected.
Nothing at all happens on double-click	The packaged build has no console. Look in `~/LogRunner/logs/start.log`.

6.3 Log and data

Situation	Meaning and remedy
"Confirmation does not match: n instead of m."	When deleting the entire log the wrong number was typed – or the window showed an outdated state. Nothing was deleted.
"Backup could not be created – nothing was deleted."	Full disk or a read-only folder. Without a safety net nothing is deleted.
"Not logged – the callsign field holds a search pattern"	The callsign field contains a `?` or `*`. A callsign nobody holds would go to Wavelog, LoTW and into the Cabrillo file and could not be recalled from there.
"Not logged – Ctrl+Alt+Enter logs anyway"	Contest mode, and the station is already in the log for this contest.
Faulty records in the import report	Lines the ADIF parser could not read. The rest of the file was read in; the pre-import backup is named in the same report.
Old QSOs have no DXCC number	Newly logged QSOs get the entity number automatically. For existing data from an import this does not apply retroactively.

6.4 Wavelog sync

Message	Meaning and remedy
"No connection to Wavelog – the QSOs stay in the queue …"	Not an error in the strict sense. The QSOs stay and go along with the next *Upload now*. **It does not happen by itself** – since 0.9.3b45 LogRunner only syncs when the button is pressed.
"The token lacks scopes: …"	Connection and station list work, uploading will fail. Add `qso:read` and `qso:write` to the token in Wavelog.
"This needs API v2 – a token with the prefix `wl2_`."	*Send changes* only works with API v2; the old API can no longer change a QSO already uploaded.
"n QSOs not matched: Wavelog has no unique entry for them."	Deleted there, or there are two identical ones.
"n QSOs keep waiting: their location has no Wavelog logbook assigned."	Under *Station locations* assign a Wavelog location to the location concerned.
"The account has n locations; the others are only fetched when …"	The same in the other direction.

6.5 Callsign lookup

The test button in the panel names the reason in plain words:

Reason	Meaning
No service is switched on.	All three services are off
No callsign entered.	The test field is empty
Not a US callsign – and callook.info is the only service switched on.	callook only knows K, N, W and AA–AL
None of the services asked knows this callsign.	Answer received, but no match
The service cannot be reached.	No network or timeout exceeded
User name or password was rejected.	Check the credentials
The lookup failed.	Unexpected answer from the service

If half the fields are missing with QRZ.com, it is usually the subscription: the XML interface only serves the complete record to accounts with *XML Logbook Data*. The reason appears with the test lookup.

6.6 DX cluster

Message	Meaning and remedy
"No login callsign: neither here nor at the station location."	Without a callsign every node refuses the connection. Enter one in the cluster dialog or in the active station location.
"No server entered"	Tick at least one node in the cluster settings.
"Not connected."	The node cannot be reached or has dropped the connection. Further attempts run in the background; program startup is never held up by it.
"All spots are older than the set time."	The expiry time applies. The table keeps showing the spots – it is the record; band map, map and waterfall show where something can be heard *now*.
"The statistics database cannot be opened."	`~/LogRunner/data/dxcluster.sqlite` is damaged or read-only. Only the chart stays empty; the spot list is not affected. The file can safely be deleted; it is rebuilt.

Message	Meaning and remedy
The list fills up with machine spots	Skimmer spots (RBN) are switched on. Switch them off – or enlarge the buffer and work specifically with the spotter filter `*-#`.

6.7 TRX control and waterfall

Situation	Meaning and remedy
The port cannot be opened	Another program holds it – often a still running `rigctld`, Omnirig or remote control software. A crashed predecessor process can also still hold the COM port; LogRunner explicitly releases it on exit.
"Transmit state cannot be queried"	The rig does not know command 1Ch (rigs older than about 2005). Then switch off *Query transmit state*, otherwise every query waits in vain.
Connected, but nothing arrives (Icom)	Check CI-V address and baud rate in the rig menu. On rigs with built-in USB, *CI-V USB Port* must be set to *Unlink from REMOTE*. Leave *CI-V Transceive* switched on.
"This transceiver delivers no spectrum over CI-V."	No band scope, or the connection runs through the `[REMOTE]` jack. The waterfall only works over USB or LAN.
"The transceiver is outside the displayed range."	VFO and displayed span do not match.
A click on a spot does not tune the rig	*Send a clicked DX spot to the transceiver* is off.
No or hardly any DX spots in the waterfall	The displayed span is too narrow. In CENTER it is VFO ± *span* – at ±5 kHz almost every spot lies outside and is therefore not drawn; only what is in the picture can be clicked there. Widen the span or set the band scope to FIX with suitable edges.
The *Rig controls* bar is empty or missing	It only exists with Icom CI-V. If it says *This transceiver provides none of these settings over CI-V*, the rig rejected every one of the commands – then check CI-V address and baud rate.
The display freezes, but the Remote Utility shows *connected*	The virtual COM port does not carry what is supposed to run over it – typically after raising the baud rate with the waterfall switched on and a short polling interval. For the sequence of steps see below.
Nothing new in the log for minutes	Neither a good sign nor a bad one: LogRunner writes a repeating error to the log only the **first** time, otherwise two hundred identical lines would be there. It keeps trying every five seconds. What currently applies is shown in the CAT panel, not in the file.

If the connection via the Icom Remote Utility freezes, in this order:

1. In *Tools* ▸ *TRX control* **switch off** the CAT connection. That stops the attempts every five seconds, and LogRunner releases the port.
2. **Quit and restart the Icom Remote Utility.** That sets up the virtual COM port again; by then it may even have disappeared from the list of ports entirely.
3. Set the **polling interval to 300 ms**, switch CAT back on and check whether the frequency follows.
4. Only then switch the **waterfall** back on – that way you can tell which of the two parts overloaded the line.

In the log (`~/LogRunner/logs/logrunner.log`) three messages have to be told apart; they stand for three different states:

Message	State
Keine Antwort von CI-V-Adresse …h an COM… (no answer from CI-V address)	The port is open, the other side is silent – wrong address or baud rate, or the way there no longer carries.

Message	State
… *lässt sich nicht öffnen: Das System kann die angegebene Datei nicht finden* (cannot be opened: file not found)	The port does **not** exist right now. With a virtual port that means: the program behind it has unregistered it.
… *lässt sich nicht öffnen: Zugriff verweigert* (cannot be opened: access denied)	It exists, but another program holds it – possibly a crashed predecessor process of LogRunner itself. If restarting the Remote Utility does not help, restarting the computer does.

6.8 WSJT-X link

Situation	Meaning and remedy
"WSJT-X link could not start" (log)	The UDP port is taken – usually by JTAlert, GridTracker or a LogRunner instance still running. Solution: use the same multicast address (e.g. `224.0.0.1`) in all programs involved.
Nothing arrives	In WSJT-X under *File* ▸ *Settings* ▸ *Reporting* check that "UDP Server" points to the same address and port. "Accept UDP requests" is not needed – LogRunner only listens.
Callsign or locator differ	Your own callsign and locator come from the active station location, not from WSJT-X. If they differ, the event list says so; logging happens under the location.

6.9 Screen capture

Message	Meaning and remedy
"ffmpeg is not present on this computer."	The *Get ffmpeg* button downloads it (about 110 MB, once). On Linux/macOS install it through the package manager and enter the path.
"No camera found. Is the capture card plugged in?"	Check the card, then *Search again*.
"This camera does not deliver MJPEG – the picture is re-encoded."	Not an error, but CPU time. If possible choose an MJPEG mode.
"Connection to the picture lost"	The picture stream broke off. *Try again*. The counter *n restarts* shows how often that has happened.
The picture stutters over Wi-Fi	Switch to *Re-encode* and above all reduce width and outgoing frame rate – that has more effect than the quality level.

6.10 LiveLink

Situation	Meaning and remedy
The hub is not found	mDNS is swallowed on the network. Enter the hub address by hand – it beats the search. As a fallback the last address found applies.
The firewall asks	Normal at the first start of a hub. On Linux open the port (`ufw allow 8765/tcp`) and provide the `avahi-daemon`.
Buttons are greyed out on the client	A mirroring window does not log by itself and holds no cluster connection. Logging and connecting happen on the hub; each window sets filters and alarms for itself.
A companion window shows an empty window	The mask entered is unknown. LogRunner then falls back to the whole window; valid identifiers are listed in section 4.2.

6.11 When nothing helps

1. Close LogRunner.
2. Save `~/LogRunner/logs/logrunner.log` and `start.log`.
3. In *Help ▸ About* copy the system information to the clipboard (at the next start) and attach both to the bug report.
4. To reset the user interface it is enough to rename `config.toml` – the log itself stays untouched. **The credentials must be entered again afterwards.**

Appendix: Help and support

If LogRunner makes your work in the shack or on the road easier, I am happy about a virtual coffee. Your support goes straight into further development, new features and prototypes.

- Ko-fi: https://ko-fi.com/dl1egl
- PayPal: https://www.paypal.com/donate/?hosted_button_id=3CPP6CJSENUTG

Both are also in the program: *Help* ▸ → *About LogRunner*, section *Help / Support*. The buttons there open the page in the system browser – that is where login and payment methods live.

Appendix: Data and licences

LogRunner itself is provided under an end-user licence agreement; it lies next to `LogRunner.exe` as `LICENSE.txt` and is presented once in the start window at the first start and after every new version (see 2.7). The terms of the included third-party software and data are in `THIRD-PARTY-NOTICES.txt` next to it.

LogRunner ships the following data:

Data set	Source and licence
Coastlines, borders, rivers	Natural Earth 1:10m (public domain)
CQ and ITU zones	`hamradio-zones-geojson`, MIT, © 2024 HB9HIL
Place data	GeoNames, CC BY 4.0
DXCC prefixes (`cty.dat`)	AD1C, country-files.com

POTA, SOTA, IOTA and Super Check Partial lists are not shipped but fetched from the respective source at the press of a button.