Dr. Lauer IT
00011010 00001100 01000111



Dr. Lauer IT

CANcorder

User manual

Record, understand and document CAN traffic

Product overview and operating instructions

Version 1.3.1247

---

https://www.vanille.de/apps/cancorder/

Contents

Meet CANcorder 3

CANcorder Help 4

Connecting to CAN Hardware 5

File Formats 11

CAN Frame Table 15

ISO-TP Messages 17

Filtering 20

Protocols Quick Reference 23

Protocol Hints 26

Plugins 30

J1939 Dashboard 35

Quality Test 37

Utilities 37

Troubleshooting 38

Meet CANcorder

CANcorder turns recorded bus traffic into something you can inspect: individual frames, complete diagnostic messages and their meaning. It is a desktop tool for people who develop, test or troubleshoot vehicle electronics and want to understand what happened on a CAN bus or a K-Line connection.

From capture to explanation

Start with a saved log or connect supported hardware. Inspect the original frames, narrow the view with filters, then follow the reconstructed transport messages and protocol hints. CANcorder keeps the raw data and its interpretation close together, so you can check a decoded value against the bytes behind it.

Typical tasks include finding an ECU's response to a diagnostic request, examining a multi-frame transfer, comparing message timing and watching decoded J1939 values. Plugins let you add interpretations for proprietary messages. The following chapters describe the supported protocols and their limits.

Results you can share

Save captures for later analysis or export a PDF report with selected frames, transport messages, protocol hints and response-time statistics. Reports can include a title and notes, making a capture easier to discuss with colleagues or attach to a test record. See File formats for the available formats and report options.

Try your workflow before choosing hardware

You can explore CANcorder with an existing log or the bundled example log; no adapter is needed for offline analysis. For live capture, the connection options depend on your operating system and adapter: TCP loggers, Windows J2534 PassThru interfaces, Linux SocketCAN and selected native macOS USB adapters. Check Connecting to hardware for device support, K-Line limitations and access modes before choosing an interface.

About this manual

This manual covers CANcorder 1.3.1247. It contains the same chapters as the in-app help, including the Quick start. You can read it from the beginning or use the contents and chapter links to look up a task. For current downloads and licensing information, visit the CANcorder product page.

CANcorder Help

CANcorder is a CAN bus and K-Line logger and analyzer for automotive diagnostics. It connects to CAN hardware via TCP, J2534 PassThru (Windows), SocketCAN (Linux), or select USB adapters on macOS (OpenPort 2.0, TouCAN, gs_usb-compatible devices). J2534 also supports K-Line protocol capture. CANcorder records and analyzes CAN and K-Line frames, reassembles multi-frame messages across seven transport protocols (ISO-TP, ISO-TP CEA, SAE TP2.0, K-Line TP, J1939, CCP, XCP), and interprets UDS/KWP/OBD-II protocols.

Quick Start

1. **Connect to hardware**: Use Connection menu to select a discovered logger, or enter IP/port manually
2. **Or load a log file**: Use File → Open Log or drag a file onto the window
3. **Enable ISO-TP**: Check "ISO-TP Decode" in the Filter panel to see reassembled messages
4. **Enable Hints**: Check "Hints" to see protocol interpretation
5. **Load example log**: Use Help → Load Example Log to explore OBD-II, UDS, KWP, and the example plugin hints

Help Topics

Use **Open PDF manual** in this help window to open the installed handbook in your system PDF viewer. Both English and German editions are included with CANcorder; the button follows the language of the app. You can read or print the handbook without an internet connection.

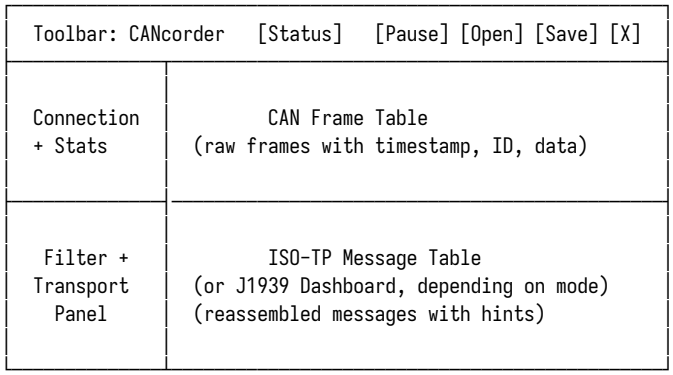
- Connecting to Hardware — TCP, J2534 (Windows, CAN + K-Line), SocketCAN (Linux), and USB (macOS) connection setup, sending CAN frames
- File Formats — Supported log formats (CL1000, candump, candump CSV, ECUconnect, PCAPNG)
- CAN Frame Table — Understanding the raw frame display
- ISO-TP Messages — Multi-frame message reassembly (ISO-TP, SAE TP2.0, K-Line TP, J1939, CCP, XCP)
- Filtering — Filter by ID, DLC, ECU, and message type
- Protocols Quick Reference — OBD-II, UDS, KWP2000, ISO-TP, J1939, CCP, and XCP cheat sheet
- Protocol Hints — UDS, OBD-II, KWP2000, and J1939 interpretation
- Plugins — Extend protocol support with custom plugins
- J1939 Dashboard — ECU-grouped live field dashboard for varying J1939 values
- Quality Test — Understand received, missing, duplicates, reordered frames, latency, and jitter
- Utilities — Live scanning helpers for macOS and Linux
- Troubleshooting — Connection, decoding, and performance tips

Interface Overview

The app is split into a left sidebar and a main analysis area:

- **Connection panel** — Connection mode, adapter settings, connect/disconnect actions, recording ID filter, and integrated live statistics
- **Quality Test panel** — Optional packet quality measurement tools

- **Filter panel** — ECU, ID, DLC, and table visibility filters
- **Transport panel** — Transport decoder selection, addressing mode, hints, and smart filtering
- **Main view** — CAN Frame Table plus ISO-TP/transport messages, or the J1939 Dashboard when enabled



Keyboard & Mouse

- Click frame/message: Selects the frame/message and copies payload data to the clipboard
- Shift+Click frame: Sets the measure point used for delta time display
- Drag file to window: Opens the log file
- Click hint row: Expands or collapses hint details
- Cmd/Ctrl+F: Open hex search
- Enter/Shift+Enter: Navigate to next/previous search result
- Escape: Close search

Version

CANcorder 1.3.1247

Connecting to CAN Hardware

CANcorder connects to CAN hardware via TCP, J2534 PassThru (Windows), SocketCAN (Linux), or select USB adapters on macOS. J2534 also supports K-Line protocol capture. TCP adapters must expose a TCP server that streams CAN frames in the expected binary format.

Connection Panel

The Connection panel is the top section of the left sidebar. It contains the connection controls, the recording ID filter, and the live statistics block that used to be shown in a separate stats panel.

Connection Modes

- **TCP** (all platforms) — Connect to a logger or adapter that streams frames over TCP.
- **J2534** (Windows) — Use installed J2534 PassThru DLLs. Supports CAN and K-Line protocols.
- **SocketCAN** (Linux) — Connect to a local SocketCAN interface (e.g., `can0`, `vcan0`).

- **USB** (macOS) — Device-by-device USB support. Currently: PEAK PCAN-USB FD and PCAN-USB Pro FD, Tactrix OpenPort 2.0, Rusoku TouCAN, and gs_usb-compatible devices (CANable, Innomaker USB2CAN, candleLight, etc.).

Note: The J2534/SocketCAN/USB tabs only appear on supported platforms.

Fields (TCP)

Field	Description	Default
Host	IP address or hostname of the CAN adapter	(blank)
Port	TCP port number	(blank)

Access Mode (USB / SocketCAN / J2534)

Non-TCP connection modes offer an access mode selector:

- **Normal** — Read and write. Enables the Send Frames button after connecting.
- **Listen-only** — Receive frames without transmitting. The adapter does not acknowledge frames on the bus, making it invisible to other nodes. Send Frames is not available.

Buttons

- **Connect** — Establish connection to the adapter
- **Disconnect** — Stop receiving frames and close connection
- **Rescan** (J2534/USB) — Refresh the detected device list
- **Send Frames** (green) — Open the send dialog (normal mode only, non-TCP)

Connected State

When a USB, SocketCAN, or J2534 session is active, the configuration fields (device, protocol, bitrate, etc.) collapse to a compact summary showing the device name, protocol, and baud rate. The Disconnect button and (in normal mode) Send Frames button remain visible.

Integrated Statistics

The bottom of the Connection panel shows live session statistics after the app has connected at least once:

- **Frame rate** — Current receive rate in frames per second
- **Busload** — Shown when the active adapter reports enough timing information to estimate it
- **Total frames** — Total frames received in the current session
- **Data received** — Approximate amount of raw capture data received
- **Streaming status** — Animated green indicator while live data is arriving

Status Indicators

The **CAN controller** section shows `error-active`, `error-warning`, `error-passive`, `bus-off`, `stopped` or `sleeping` separately from connection status. A connected adapter may still be bus-off. Supported counters include TEC/REC, observed error reports, ACK and protocol errors, bus-off transitions and queue overflows/drops. Missing counters are omitted; a stale sample keeps the last state and is labelled explicitly.

SocketCAN, gs_usb, PEAK and TouCAN provide the measurements supported by their driver and firmware. Virtual CAN, OpenPort, J2534 and the logger-only TCP stream have no physical controller health data. These values describe the attached controller, not every bus node. They cannot measure termination, and `error-active` alone does not prove successful communication. On a two-node bench, check the bitrate, one 120 Ω terminator at each end and an acknowledging node in normal mode; listen-only nodes do not ACK.

CAN reception is shown separately from controller health. Until a CAN frame has been received, reception remains unconfirmed: a quiet bus and a missing physical bus connection cannot yet be distinguished. Check the bus connection, selected channel, bitrate and active peer.

- **Connected to [host]:[port]** — Active connection
- **Error message (red)** — Connection failed with reason

J2534 PassThru (Windows)

Click the device selector to open the device picker dialog. Each device card shows:

- **Name** and **vendor**
- **Architecture badge** (32-bit / 64-bit) in indigo
- **API version badge** (v04.04 / v05.00) — green for v5.0 devices
- **Supported protocols** as colored pills (CAN, ISO-TP, ISO 9141, ISO 14230, CAN-FD, etc.)
- **Availability status** — unavailable devices (missing DLL) are grayed out with a reason

After selecting a device, choose a protocol (CAN or K-Line) and connect at the desired baud rate.

CANcorder scans both v4.04 and v5.00 J2534 registry entries and uses an out-of-process bridge to load driver DLLs. This means:

- **All drivers work** regardless of architecture — 32-bit and 64-bit DLLs are both supported.
- **Driver crashes are isolated** — a buggy DLL crash kills only the bridge process, not CANcorder.

CAN Protocol

The channel is opened with a pass-all filter and `CAN_ID_BOTH`, so both standard (11-bit) and extended (29-bit) frames are received.

K-Line Protocol

Select **K-Line** in the protocol dropdown to capture K-Line serial bus traffic. K-Line TP transport decoding is automatically enabled on connect. The default baud rate is 10400 (ISO 14230). Enable the keepalive option if the ECU requires periodic TesterPresent messages to stay in a diagnostic session.

On macOS, native OpenPort K-Line support is initialized-bus-only. CANcorder can listen to and write raw K-Line frames after another tool, ECU wakeup sequence, or bench setup has already initialized the K-Line session. Fast init, five-baud init, automatic protocol detection, and keepalive are only advertised for backends that explicitly support them, such as Windows J2534.

SocketCAN (Linux)

Use the **SocketCAN** tab to select an interface (like `can0` or `vcan0`) and set the bitrate. CANcorder uses the system SocketCAN stack to receive frames locally.

- If no interfaces appear, create or bring one up with your normal `ip link`/`can-utils` workflow

- The bitrate field should match the interface configuration

USB Adapters (macOS)

Use the **USB** tab to select a supported adapter. Current support is device-by-device:

- **PEAK PCAN-USB FD** — Single-channel CAN-FD adapter (IPEH-004022)
- **PEAK PCAN-USB Pro FD** — Dual-channel CAN-FD adapter (IPEH-004061). Both CAN channels appear as separate devices in the device list. CAN 1 uses DB-9 pins 7/2 (CANH/CANL), CAN 2 uses pins 4/9.
- **Tactrix OpenPort 2.0** — Classic CAN and native K-Line on macOS. Native K-Line is tested for sniffing/writing on an already initialized ISO 14230 bus at 10400 baud; K-Line initialization remains a Windows J2534 capability.
- **Rusoku TouCAN**
- **gs_usb-compatible devices** — CANable, CANable 2.0, Innomaker USB2CAN, candleLight, CANtact, and other adapters running candleLight/gs_usb firmware
- Requires libusb (`brew install libusb`)
- Connect and choose a baud rate; CAN frames stream into the table like TCP
- For gs_usb adapters that support CAN-FD, select **CAN-FD** and set both nominal baud rate and data baud rate (for example `500000` / `2000000`)

Sending CAN Frames

When connected via USB (macOS), SocketCAN (Linux), or J2534 (Windows) in **normal** (non-listen-only) mode, a green **Send Frames** button appears in the connection panel. Clicking it opens the Send Frames dialog. TCP connections are receive-only and do not support sending.

Sent frames are echoed back into the frame table so you can see both your requests and the device responses.

Note (J2534): The timestamp of a sent frame may show as 0.000 if no frames have been received yet. J2534 devices only provide timestamps on received frames and offer no API to query the current device clock. Once the first frame is received, subsequent sent frame timestamps will be correct.

Single Tab

Send individual CAN frames interactively.

Field	Description
CAN ID (hex)	Arbitration ID (e.g. 7DF). Auto-uppercased.
Std / Ext toggle	Standard (11-bit) or Extended (29-bit) ID
Data (hex bytes)	Payload bytes, space-separated (e.g. 02 01
	00). Auto-formatted with spaces and
	uppercasing as you type.
Pad to DLC	Zero-pad (or custom byte) to selected DLC.
	Default: pad to 8 with 0xAA.
CAN-FD frame	Send as FD frame (only if adapter supports

Field	Description
	CAN-FD)

After sending, the frame appears in a **History** list (last 20 entries). Click the resend icon to re-transmit a previous frame.

Sequence Tab

Send multiple frames automatically using one of three modes:

- **Repeat** — Send the same frame N times at the configured rate
- **Sweep** — Iterate over a CAN ID range (start \u2192 end, with configurable step) sending a fixed data payload
- **Custom List** — Enter one frame per line (`ID DATA`, e.g. `7DF 02 01 00`). Optionally loop continuously.

All sequence modes share a **rate control**:

- **FPS mode** — Set target frames per second (e.g. 100 fps \u2192 10 ms interval)
- **Interval mode** — Set milliseconds between frames (e.g. 10 ms \u2192 100 fps)

A progress bar shows send progress. Click **Stop** to cancel a running sequence.

Settings Persistence

All dialog settings (IDs, data, padding, rate, sequence mode, custom list, and send history) are saved automatically and restored on next launch.

Zeroconf (mDNS/Bonjour) Discovery (TCP)

CANcorder automatically discovers ECUconnect loggers that advertise the `_ecuconnect-log._tcp.local.` service over Zeroconf (mDNS/Bonjour).

- Discovered loggers show up in the Connection menu and in the Connect dialog
- Selecting a logger fills in host/port and connects immediately
- If no services are found, connect manually using Host and Port

Note: Some networks block multicast DNS. If discovery does not find your logger, ensure mDNS is enabled on the logger and not filtered by the network.

Live Streaming

When connected, CANcorder receives CAN frames in real-time:

- The header shows "Recording" with a red indicator
- Frames appear in the CAN Frame Table as they arrive
- The Connection panel statistics section shows live frame rate and totals
- Auto-scroll keeps the newest frames visible

Pause/Resume

During live streaming, a Pause button appears in the header:

- **Yellow (Pause)** — Click to stop auto-scrolling; you can scroll through history
- **Green (Play)** — Click to resume auto-scrolling to newest frames

Pausing does not stop frame reception — frames continue to be recorded.

Frame Storage

CANcorder keeps all received frames in memory for fast access. Memory usage grows with capture length, so long sessions can consume significant RAM. For extended logging, a future option is to stream frames to disk (or use a rolling buffer) to keep memory usage stable.

Disconnecting

When you click Disconnect with recorded data, a confirmation dialog appears:

> "You have recorded data. Are you sure you want to disconnect?"

This prevents accidental data loss.

Loading Files While Connected

If you try to open a log file while connected, CANcorder will ask:

> "Loading a file will disconnect from the logger. Continue?"

Confirm to disconnect and load the file, or cancel to stay connected.

Wire Protocol

CANcorder expects frames in this binary format:

Bytes	Field	Format
0-7	Timestamp	64-bit big-endian microseconds
8-11	CAN ID	32-bit big-endian
12	Flags	bit0=EXT, bit1=FD, bit2=BRS, bit3=ESI
13	DLC	Data length (0-64)
14+	Data	Raw bytes

Legacy senders that only set byte 12 to `0` (standard) or `1` (extended) are still supported.

File Formats

CANcorder can import log files in several formats. The format is auto-detected based on file content.

Opening Files

- **Menu**: File → Open Log
- **Toolbar**: Click the folder icon
- **Drag & Drop**: Drag a file onto the window

Supported extensions: `.txt`, `.log`, `.pcapng`, `.csv`

An opened log reloads automatically after an external edit or replacement has settled. Display filters remain active. If the file is temporarily missing or cannot be parsed, the current log stays visible with a status message; CANcorder retries automatically. File reloading pauses during a live capture.

PDF reports

Choose **File** → **Export as PDF…** (Cmd/Ctrl+Alt+E), or the PDF button in the toolbar. Choose **Messages** for raw bus frames or **Transport** for decoded messages, then **Current filters** or **Everything**. Transport uses the current decoder and hint settings; enable Transport Decode first if no decoded messages are available.

Current filters uses the same ID, ECU, length and transport visibility rules as the corresponding table. Search highlighting and row selection do not restrict the export. The report uses the log and filter settings from when the dialog opened; live recording can continue.

The color PDF includes the selection, original record numbers, microsecond timing, complete hex/ASCII payloads and page numbers. Large payloads continue across pages. Barlow and Iosevka fonts are embedded, so the report looks the same on another computer. Export works offline and can be cancelled during creation.

The report uses A4 portrait. The front matter has its own page, with the CANcorder icon beside the title; the table always starts on a new page. Long payloads use eight bytes per line, with offsets and ASCII. The footer includes version, license email when licensed, copyright and website. Colored symbols replace request/response status text: right arrow for requests, check for positive responses, cross for negative responses, clock for pending responses, and left arrow for other responses. The front matter includes a labeled legend. Compact rows keep address pairs together, with payloads in a separate ink color from the interpretation.

Response-time statistics show best, average, worst, jitter (population standard deviation) and P95, plus a breakdown by ECU when several responders are present. Only complete, unambiguous OBD/UDS/KWP pairs with both ends in the export count. Timing runs from the last request frame to the first final-response frame, or consistently from request start when raw request frames are unavailable; the report states the basis. NRC 0x78 keeps the request pending. Sources without recorded timestamps, such as minimal candump CSV, have no timing statistics.

CL1000 Format

The CL1000 format is used by CSS Electronics CAN loggers. CANcorder uses this format for both import and export.

Structure

```
# Logger type: CAN-Logger-App
# HW rev: 1.0
# FW rev: 1.0
# Logger ID: app0001
# Time: 20231015T143052
# Bit-rate: 500000
Timestamp;Type;ID;Data
01T000000000;0;7E0;0322F190
01T000000050;0;7E8;1014622F190574F
```

Fields

Column	Description
Timestamp	`DDTHHMMSSmmm` — days, hours, minutes, seconds, milliseconds
Type	`0` = standard (11-bit), `1` = extended (29-bit)
ID	CAN identifier in hex (no prefix)
Data	Payload bytes in hex (concatenated)

Metadata Displayed

- Logger type, ID
- Hardware/firmware revision
- Capture time
- Bit rate
- Session and split numbers
- Silent/cyclic mode flags

candump Format

The candump format is produced by Linux can-utils (`candump -L`).

Structure

```
(1766060076.353543) can1 08FF0001#0200FFFFCF00AC7D
(1766060076.353755) can1 08FF0101#0100A25F000091B2
```

CAN FD frames use a second # followed by a flags nibble (bit 0 = BRS, bit 1 = ESI). CANcorder reads that form and marks the frame accordingly:

```
(1766060076.353543) can1 123##1AABBCCDD
```

ECUmulator++ writes this format with `--can-log`, using the interface column to record direction: `<channel>rx` for frames entering the emulation and `<channel>tx` for frames leaving it.

Fields

Part	Description
`(timestamp)`	Seconds.microseconds; may be negative if the logger had no real-time clock set
`interface`	CAN interface name (e.g., `can0`, `vcan0`)

Part	Description
`id`	CAN ID in hex (>3 digits = extended frame)
`#`	Separator
`data`	Payload bytes in hex (concatenated)

Metadata Displayed

- Interface name (shown as purple badge)
- Frame count
- Capture start/end times
- Duration

Minimal candump CSV

Some tools export a CSV with only the arbitration ID and payload bytes (no timestamps).

Structure

```
ID,Data Bytes
000007C0,06 06 FF 02 05 FF FF FF
00000130,F3 FF FF FF FF
```

Fields

Column	Description
ID	CAN identifier in hex (no prefix)
Data Bytes	Payload bytes as space-separated hex pairs

Notes

- Timestamps are not present, so CANcorder synthesizes monotonically increasing timestamps for display.
- Capture start/end times and duration are not shown.

ECUconnect Format

Logs from the ECUconnect iOS app contain ISO-TP messages with rich metadata.

Structure

```
#META:timestamp=12:34:56.789
#META:vin=WBADP5555E123456
#META:ecu=DME
00:00:00.000,7E0,0322F190
00:00:00.050,7E8,1014622F190574F
```

Metadata Displayed

- VIN (prominent display)
- ECU name
- App and adapter info

- Tester and ECU addresses
- Task description
- Message and error counts

Special Handling

ECUconnect logs are treated like PCAPNG files:

- Only the ISO-TP table is shown (no raw CAN frames)
- ISO-TP decode and hints are auto-enabled

PCAPNG Format

Packet capture files containing DoIP or HSFZ Ethernet traffic.

Supported Protocols

- **HSFZ** — ISO 13400-2 Ethernet header framing
- **DoIP** — ISO 13400-1 Diagnostic over IP

Metadata Displayed

- Protocol type (HSFZ/DoIP/mixed)
- VIN (if present in responses)
- Packet and message counts
- Capture start/end times
- IP addresses (source and destination)
- ECU addresses
- Network interfaces

Special Handling

- Parsed by native Rust backend for performance
- Only ISO-TP messages are shown (no raw frames)
- ISO-TP decode and hints are auto-enabled

Metadata Dialog

Click the info button (`i`) next to a loaded filename to view metadata:

- Format-specific header with colored badge
- VIN display (for ECUconnect/PCAPNG)
- Grid of metadata fields
- Address lists and interface info

Format Detection

CANcorder detects formats in this order:

1. **PCAPNG** — by `.pcapng` extension
2. **ECUconnect** — by `#META:` prefix in content

3. **Minimal candump CSV** — by `ID,Data Bytes` header or `id,data` CSV lines
4. **candump** — by `(timestamp) interface id#data` pattern
5. **CL1000** — fallback for other text files

CAN Frame Table

The CAN Frame Table displays raw CAN frames as they are received or loaded from a file.

Columns

Column	Description
#	Frame index (1-based)
Time	Elapsed time in seconds from the first frame
ID	CAN identifier in hex
Ext	`Y` for extended (29-bit), `N` for standard (11-bit)
DLC	Data Length Code — number of payload bytes
Data	Payload bytes in hex, space-separated
ASCII	Printable ASCII representation (`.` for non-printable)

Color Coding

ID Column

- **Green** — Standard frame (11-bit ID, 0x000–0x7FF)
- **Orange** — Extended frame (29-bit ID)

Row Selection

- **Blue bar** — Currently selected frame
- **Emerald bar** — Measure point (for time delta)

Interactions

Click

- Selects the frame
- Copies frame data to clipboard as hex (e.g., `03 22 F1 90`)
- If ISO-TP is enabled, selects the corresponding ISO-TP message

Shift+Click

- Sets a measure point for time delta calculation
- The time column shows **Δt** — the time difference from the measure point

- Click normally to clear the measure point

Scroll

- Mouse wheel scrolls through frames
- During live streaming with auto-scroll enabled, new frames appear at bottom
- Pause to scroll freely through history

Frame Counter

The header shows: `1,234 / 5,000 frames`

- First number: frames matching current filter
- Second number: total frames loaded

When no frames match the filter: "No frames match filter"

Time Display

Normal Mode

Shows elapsed time from first frame:

- `0.000` — first frame
- `1.234` — 1.234 seconds after first frame

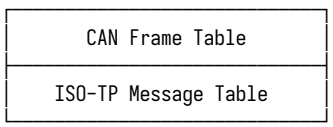
Time Delta Mode (Shift+Click)

Shows Δt (delta time) from the measure point:

- `Δt: 0.050` — 50ms after measure point
- `Δt: -0.100` — 100ms before measure point

Split View

When ISO-TP Decode is enabled, the display splits:



The split view stacks the CAN Frame Table above the ISO-TP Message Table, separated by a draggable divider.

- Drag the divider to adjust the split ratio (15%–85%)
- The divider highlights when hovering or dragging

Performance

The table uses virtualized scrolling to handle large datasets efficiently:

- Only visible rows are rendered
- Smooth scrolling through 10,000+ frames
- Memory use scales with total frames kept in RAM; long captures can consume more memory

- Disk-backed capture or rolling buffers can mitigate memory growth for long sessions

ISO-TP Messages

ISO-TP (ISO 15765-2) is a transport protocol that allows messages larger than 8 bytes to be sent over CAN. CANcorder reassembles these multi-frame messages automatically.

Enabling ISO-TP Decode

In the Filter panel, check **ISO-TP Decode** to:

- Reassemble multi-frame messages
- Show the ISO-TP Message Table
- Enable protocol hint interpretation

Addressing Modes

Select the addressing mode in the Filter panel dropdown:

Mode	Description
ISO-TP	Standard ISO-TP with 11-bit or 29-bit CAN IDs
ISO-TP (CEA)	Uses first data byte as extended address
SAE TP2.0	Volkswagen Group TP 2.0 transport protocol
K-Line TP	K-Line serial bus transport protocol
J1939	SAE J1939 transport (BAM and RTS/CTS multi-packet)
CCP	CAN Calibration Protocol (ASAM MCD-1 / CCP 2.1)
XCP	Universal Measurement and Calibration Protocol on CAN

Extended addressing (CEA) is used when multiple ECUs share the same CAN ID. K-Line TP is automatically selected when connecting via J2534 with the K-Line protocol.

J1939 Transport

J1939 uses 29-bit extended CAN IDs with a structured format encoding priority, PGN (Parameter Group Number), source address, and destination address. CANcorder decodes J1939 transport-layer messages:

- **BAM (Broadcast Announce Message)**: Sender broadcasts a multi-packet message without handshaking. All nodes on the bus receive the data.
- **RTS/CTS (Request to Send / Clear to Send)**: Point-to-point multi-packet transfer with flow control between sender and receiver.
- **TP.CM / TP.DT**: Transport Protocol Connection Management and Data Transfer frames carry the session control and payload data respectively.

CANcorder reassembles fragmented J1939 payloads and displays them as single messages with PGN, priority, and address metadata.

CCP (CAN Calibration Protocol)

CCP is used for ECU calibration and measurement via CAN. Communication uses fixed CAN IDs with a command/response pattern:

- **CRO (Command Receive Object)**: Tester-to-ECU commands (CONNECT, UPLOAD, DOWNLOAD, SET_MTA, etc.)
- **DTO (Data Transfer Object)**: ECU-to-tester responses, including error codes and data
- **DAQ (Data Acquisition)**: Periodic measurement data packets from the ECU

CANcorder pairs commands with their responses using the CTR (counter) byte and displays decoded command names and error codes.

XCP (Universal Measurement and Calibration Protocol)

XCP is the successor to CCP, supporting higher throughput and more features. On CAN, it also uses fixed CAN IDs:

- **CMD**: Tester-to-ECU commands (CONNECT, GET_STATUS, UPLOAD, DOWNLOAD, DAQ configuration, etc.)
- **RES/ERR**: ECU-to-tester responses (positive or error)
- **DAQ/STIM**: Periodic measurement and stimulation data

CANcorder pairs XCP commands with responses using timing heuristics and displays decoded command names and error codes.

ISO-TP Frame Types

Type	Name	Purpose
SF	Single Frame	Message fits in one frame (≤7 bytes)
FF	First Frame	First frame of multi-frame message
CF	Consecutive Frame	Continuation of multi-frame message
FC	Flow Control	Receiver's response to FF (timing parameters)

Message Table Columns

Column	Description
#	Message index (1-based)
Time	Elapsed time in seconds
TX ID	Transmitter CAN ID (orange)
RX ID	Receiver CAN ID (cyan)
CEA	Extended address byte (pink), or – if not used
Dir	Direction: `REQ` (blue) or `RSP` (green)

Column	Description
Len	Total message length in bytes
Data	Payload in hex

Direction Detection

CANcorder determines request vs response by:

- **OBD-II pattern**: IDs 0x7E0–0x7E7 are requests, 0x7E8–0x7EF are responses
- **UDS pattern**: Response ID = Request ID + 8 (for functional addressing)
- **Request SIDs**: Messages starting with known request service IDs

Data Display

Short Messages (≤40 bytes)

Data is shown inline in the Data column.

Long Messages (>40 bytes)

- Data is truncated with a ▶ indicator
- Click ▶ to expand and show all bytes
- Click again to collapse

Row Highlighting

Color	Meaning
Light blue background	Request message
Light green background	Response message
Darker blue	Selected message

Message Counter

The header shows: `234 / 1,000 messages`

- First number: messages matching current filter
- Second number: total messages

Selecting Messages

Click

- Selects the message
- Copies payload to clipboard as hex
- If CAN frames are visible, highlights the first frame of this message

Frame Correlation

When you select a CAN frame, CANcorder finds and selects the ISO-TP message containing that frame. This works even for Flow Control frames that aren't part of the message payload.

Transfer Data Export

For UDS TransferData (0x36) sequences, a "Save Transfer Data" button appears:

- Shows transfer direction (download/upload)
- Shows byte count
- Shows target address if available
- Saves reassembled binary data to a file

Filtering

The Filter panel lets you focus on specific frames and messages by ID, type, length, and more.

Hex Search

Click the magnifying glass icon in the header to search for hex sequences in frame/message data.

Usage

1. Click the search icon (turns blue when active)
2. Type a hex sequence (e.g., `22F190`)
3. Use ▲/▼ buttons or Enter/Shift+Enter to navigate results
4. Press Escape to close

Features

- **Result counter**: Shows "3 of 15" format
- **Highlighting**: Current result has amber ring, all matches have subtle amber tint
- **Scope-aware**: Only searches visible tables (CAN frames when ISO-TP is disabled, both when enabled)
- **Flexible input**: Spaces and separators are ignored (`22 F1 90` = `22F190` = `22:F1:90`)

ECU Filter

Detected ECUs appear as clickable pills at the top of the Filter panel.

Few ECUs (≤6)

Each ECU shows as an individual pill:

- **Visible**: Colored background
- **Hidden**: Gray/muted
- Click to toggle visibility

Many ECUs (>6)

A dropdown popover appears with:

- **Show all** / **Hide all** buttons
- Checkbox for each ECU
- Message count per ECU

What Counts as an ECU?

For CAN frames, each unique CAN ID is treated as an ECU. For PCAPNG/ECUconnect files, both TX and RX IDs are counted.

ID Filter

Enter CAN IDs to show in the **ID Filter (hex)** field.

Patterns

Pattern	Matches
`7E0`	Exact ID 0x7E0
`7E0,7E8`	IDs 0x7E0 or 0x7E8
`7E*`	IDs starting with 0x7E (0x7E0–0x7EF)
`18DA*`	Extended IDs starting with 0x18DA

The filter supports regex patterns for advanced matching.

Frame Type

Checkbox	Effect
Standard	Show 11-bit CAN IDs (0x000–0x7FF)
Extended	Show 29-bit CAN IDs

Uncheck both to hide all frames (not useful, but allowed).

DLC Range

Filter by Data Length Code:

Field	Description	Range
Min DLC	Minimum payload length	0–64
Max DLC	Maximum payload length	0–64

Useful for filtering out specific frame sizes:

- Set Min=8, Max=8 to see only 8-byte frames
- Set Min=1 to exclude empty frames

ISO-TP Options

Control	Description
ISO-TP Decode	Enable multi-frame message reassembly
Protocol	ISO-TP, ISO-TP (CEA), SAE TP2.0, K-Line TP,
	J1939, CCP, or XCP

See ISO-TP Messages for details.

Hint Options

Control	Description
Hints (OBD2/UDS/KWP)	Show protocol interpretation
Smart filtering	Filter to request/response pairs on same ECU

See Protocol Hints for details.

Hint Type Filter

When hints are enabled, filter pills appear:

Pill	Color	Meaning
Request	Blue	UDS/OBD-II request messages
Positive	Green	Successful response messages
NRC	Red	Negative Response Codes
Busy	Orange	Server busy (0x21 NRC)

Click pills to show/hide each type.

Filter Combinations

Filters are combined with AND logic:

- A frame must match ALL active filters to be shown
- ECU filter AND ID filter AND DLC range AND frame type

Example: To see only DME responses:

1. Set ID filter to `7E8`
2. Check only "Standard"
3. Enable Hints and show only "Positive" type

Counter Display

Both tables show filtered vs total counts:

```
CAN Frames: 1,234 / 5,000 frames
ISO-TP:     234 / 1,000 messages
```

When no items match: "No frames match filter" or "No messages match filter"

Clearing Filters

To reset all filters:

1. Click "Show all" in ECU dropdown
2. Clear the ID filter field
3. Check both Standard and Extended
4. Set DLC range to 0–64
5. Show all hint types

Protocols Quick Reference

This page is a practical cheat sheet for common automotive diagnostic protocols as seen in CANcorder. It is not a complete specification.

ISO-TP Basics (ISO 15765-2)

- ISO-TP reassembles multi-frame payloads carried inside CAN frames
- Single Frame (SF) carries up to 7 bytes (or 6 in extended addressing)
- First Frame (FF) starts a multi-frame transfer, Consecutive Frames (CF) continue it
- Flow Control (FC) frames are ignored for decoding purposes in CANcorder

Example Single Frame payload:

```
CAN Data: 02 10 01
          ^^
          length = 2 bytes
Payload: 10 01
```

Example Multi-Frame payload (length 10):

```
FF: 10 0A 22 F1 90 01 02 03
CF: 21 04 05 06 07
Payload: 22 F1 90 01 02 03 04 05 06 07
```

UDS (ISO 14229) Service IDs

Common service IDs and meanings:

```
SID  Name                        Notes
0x10 DiagnosticSessionControl    Switch sessions (default, extended, programming)
0x11 ECUReset                    Reset ECU
0x22 ReadDataByIdentifier        Read DID values (e.g., VIN)
0x23 ReadMemoryByAddress         Read memory blocks
0x2E WriteDataByIdentifier       Write DID values
0x2F InputOutputControlById       IO control
0x31 RoutineControl              Start/stop routines
0x34 RequestDownload             Start download (programming)
0x35 RequestUpload               Start upload
0x36 TransferData                Data blocks
0x37 RequestTransferExit         End transfer
0x3E TesterPresent               Keep session alive
0x7F NegativeResponse            Error response (with NRC)
```

Responses typically echo the SID with 0x40 added:

```
Request 0x22 -> Positive response 0x62
Request 0x10 -> Positive response 0x50
```

OBD-II (SAE J1979) Quick Notes

- Functional request ID: 0x7DF (broadcast)
- ECU response IDs: 0x7E8-0x7EF (one per ECU)
- Service $01 = Current Data, $09 = Vehicle Information

Common PID examples:

```
Service  PID  Meaning
0x01     0C   Engine RPM
0x01     0D   Vehicle Speed
0x01     05   Coolant Temperature
0x09     02   VIN
```

KWP2000 (ISO 14230) Notes

- Older than UDS, still used in many ECUs
- Services overlap with UDS but have different IDs
- CANcorder hints attempt best-effort decoding; detailed vendor data often needs a plugin

SAE TP2.0 (VAG TP 2.0) Notes

- Volkswagen Group TP2.0 is a separate transport protocol on 11-bit CAN, not a variant of ISO-TP
- Channels are negotiated dynamically before data transfer, then payloads are carried in TPDU frames with ack and retransmit semantics
- CANcorder reassembles TP2.0 payloads and then applies the normal UDS/KWP hint layer to the decoded request/response bytes
- Typical fixed CAN IDs for channel setup are in the `0x200` to `0x2EF` range; dynamic data channels are assigned during setup

K-Line (ISO 14230 / ISO 9141) Notes

- K-Line is a single-wire serial bus used in older vehicles for diagnostic communication
- Typically runs at 10400 baud (ISO 14230), though some implementations use other rates
- CANcorder captures K-Line traffic via J2534 PassThru devices that support the ISO9141/ISO14230 protocols
- K-Line TP transport decoding reassembles request/response messages and applies UDS/KWP hint interpretation
- The keepalive option sends periodic TesterPresent messages to maintain the diagnostic session

J1939 (SAE J1939) Notes

- Heavy-duty vehicle standard using 29-bit extended CAN IDs
- CAN ID encodes: Priority (3 bits), Data Page, PF (PDU Format), PS (PDU Specific), Source Address
- PGN (Parameter Group Number) identifies the message type
- Two transport modes for payloads >8 bytes:

Mode	Control PGN	Data PGN	Description
BAM	0x00EC00	0x00EB00	Broadcast, no handshake
RTS/CTS	0x00EC00	0x00EB00	Point-to-point with flow control

TP.CM control byte values:

Byte	Name	Direction
0x10	RTS	Request (initiator)
0x11	CTS	Response (receiver grants)
0x13	EOM_ACK	Response (transfer complete)
0x20	BAM	Broadcast announcement
0xFF	ABORT	Abort transfer

CANcorder filters out ISO 15765-3 traffic (EDP=1 or PF=0xDA/0xDB) to avoid misclassifying UDS diagnostic frames as J1939.

CCP (CAN Calibration Protocol) Notes

- ASAM MCD-1 CCP, used for ECU calibration and measurement
- Fixed CAN ID pair: one for commands (CRO), one for responses (DTO)
- Commands use a counter byte (CTR) for request/response correlation

Common commands:

Code	Name	Purpose
0x01	CONNECT	Establish session
0x02	SET_MTA	Set memory transfer address
0x03	DOWNLOAD	Write to ECU memory
0x04	UPLOAD	Read from ECU memory
0x06	START_STOP	Start/stop DAQ lists
0x07	DISCONNECT	End session
0x0F	SHORT_UP	Short upload (up to 5 bytes)
0x12	GET_SEED	Security access (seed)
0x13	UNLOCK	Security access (key)

XCP (Universal Measurement and Calibration Protocol) Notes

- ASAM MCD-1 XCP, successor to CCP with higher throughput
- On CAN: fixed CAN ID pair for CMD/RES, plus optional DAQ/STIM IDs
- Commands use PID byte 0xFF..0xC0 range (descending)

Common commands:

Code	Name	Purpose
0xFF	CONNECT	Establish session
0xFE	DISCONNECT	End session
0xFD	GET_STATUS	Query session status
0xF5	UPLOAD	Read from ECU memory
0xF4	SHORT_UPLOAD	Short upload
0xF0	DOWNLOAD	Write to ECU memory
0xED	SHORT_DOWNLOAD	Short download
0xE2	SET_DAQ_PTR	Configure DAQ pointer
0xE1	WRITE_DAQ	Write DAQ list entry
0xDE	START_STOP_DAQ_LIST	Start/stop DAQ list
0xDD	START_STOP_SYNCH	Synchronised start/stop

Request/Response Patterns

- 11-bit addressing: requests around 0x7E0-0x7E7, responses 0x7E8-0x7EF
- 29-bit addressing: often 0x18DAxxYY, where source/target bytes swap between request/response

Protocol Hints

Protocol hints interpret reassembled transport payloads according to automotive diagnostic standards such as UDS, OBD-II, KWP2000, and J1939. CANcorder currently supports ISO-TP, ISO-TP (CEA), SAE TP2.0, K-Line TP, and J1939 transport reassembly for hints.

Enabling Hints

1. Check **Transport Decode** in the Filter panel
2. Select the transport protocol: **ISO-TP**, **ISO-TP (CEA)**, **SAE TP2.0**, **K-Line TP**, or **J1939**
3. Check **Hints (OBD2/UDS/KWP)**

Hint rows appear below each interpreted message in the transport table.

For **SAE TP2.0**, CANcorder also shows selected transport-layer control frames as transport rows with hints, including channel setup/ack, connection setup/ack, connection test, break, and disconnect.

For **J1939**, CANcorder shows PGN names for direct and reassembled payloads, decodes selected public SPN formulas for common PGNs, and renders ASCII-style identification payloads such as Software Identification, Component Identification, Vehicle Identification, and ECU Identification when possible.

J1939 Catalogs

CANcorder ships with a small built-in J1939 reference built from public/open-source material. For broader coverage, it also looks for an external `J1939db.json` and merges it over the built-in catalog.

Search locations:

- `AppConfig/J1939db.json`
- `AppConfig/j1939/J1939db.json`
- `~/.config/cancorder/J1939db.json`
- `~/.config/pretty_j1939/J1939db.json`

Compatible files include the `J1939db.json` format used by tools such as `pretty_j1939` and `j1939decode`.

Hint Display

Hints appear as yellow rows below messages. The examples below use `[Hint]` for the light-bulb indicator and `[Request]` for the blue request indicator:

```
[Hint] [Request] ReadDataByIdentifier: F190 (VIN)
```

Part	Meaning
Light bulb	Hint indicator
Status icon	Request/response type
Service name	Interpreted service
Details	Parameters and values

Status Icons

Icon	Meaning	Color
Blue dot	Request	Blue
✓	Positive response	Green
×	Negative response	Red
⏳	Busy (repeat request)	Orange

Expanding Hints

Click a hint row to expand details:

```
[Hint] ✓ ReadDataByIdentifier: F190 (VIN)
```

Field	Value	Raw bytes
Diagnostic Service	ReadDataByIdentifier	22
Data Identifier	F190 (VIN)	F1 90
VIN	WBADP5555E123456	57 42 41 ...

The detail grid shows:

- **Field name** — What the bytes represent
- **Value** — Interpreted value
- **Raw bytes** — Original hex data

UDS Services

CANcorder interprets these UDS services (ISO 14229):

SID	Service	Description
0x10	DiagnosticSessionControl	Switch diagnostic sessions
0x11	ECUReset	Reset the ECU
0x14	ClearDiagnosticInformation	Clear DTCs
0x19	ReadDTCInformation	Read fault codes
0x22	ReadDataByIdentifier	Read ECU data
0x27	SecurityAccess	Unlock protected functions
0x2E	WriteDataByIdentifier	Write ECU data
0x34	RequestDownload	Start memory download
0x35	RequestUpload	Start memory upload
0x36	TransferData	Transfer memory blocks
0x37	RequestTransferExit	End transfer
0x3E	TesterPresent	Keep session alive

Diagnostic Sessions (0x10)

Session	Name
0x01	Default Session
0x02	Programming Session
0x03	Extended Diagnostic Session
0x04	Safety System Diagnostic Session

Common Data Identifiers (0x22/0x2E)

DID	Description
0xF186	Active Diagnostic Session
0xF187	Vehicle Manufacturer Spare Part Number
0xF188	Vehicle Manufacturer ECU Software Number
0xF189	Vehicle Manufacturer ECU Software Version
0xF18A	System Supplier Identifier
0xF18C	ECU Serial Number
0xF190	VIN (Vehicle Identification Number)
0xF197	System Name or Engine Type

OBD-II Services

For OBD-II diagnostic requests (IDs 0x7DF, 0x7E0–0x7E7):

Mode	Description
0x01	Show current data (live PIDs)
0x02	Show freeze frame data
0x03	Show stored DTCs
0x04	Clear DTCs and freeze frame
0x05	Oxygen sensor monitoring
0x06	On-board monitoring test results
0x07	Show pending DTCs
0x08	Control on-board systems
0x09	Request vehicle information
0x0A	Permanent DTCs

Negative Responses

When an ECU rejects a request, it returns service 0x7F:

`[Hint] × Negative Response: ServiceNotSupported (0x11)`

Common NRC Codes

Code	Name	Meaning
0x10	generalReject	Request rejected
0x11	serviceNotSupported	Service not available
0x12	subFunctionNotSupported	Sub-function not available
0x13	incorrectMessageLength	Wrong data length
0x14	responseTooLong	Response exceeds buffer
0x21	busyRepeatRequest	ECU busy, try again
0x22	conditionsNotCorrect	Prerequisites not met
0x24	requestSequenceError	Wrong request order
0x31	requestOutOfRange	Parameter out of range
0x33	securityAccessDenied	Security not unlocked
0x35	invalidKey	Wrong security key
0x36	exceededNumberOfAttempts	Too many failed attempts
0x72	generalProgrammingFailure	Flash write failed
0x78	responsePending	Processing, response coming

Smart Filtering

When **Smart filtering** is enabled:

- Only shows hints for ECU pairs with both request and response
- Keeps hints for canonical diagnostic request IDs (for example `7DF`, `7E0-7E7`, `18DA*`, `18DB*`) even if no response arrives
- Filters out ambiguous broadcast messages without responses
- Reduces noise in busy CAN logs
- In **ISO-TP (CEA)**, requests may start without a known RX ID; hints can appear after the first matching response is observed
- In **SAE TP2.0**, hints are generated from the reassembled TPDU payload after TP2.0 channel traffic is decoded

This feature is available for live streams and CAN frame logs (not PCAPNG/ECUconnect).

Hint Type Filtering

Use the colored pills in the Filter panel to show/hide:

- **Request** (blue) — Outgoing diagnostic requests

- **Positive** (green) — Successful responses
- **NRC** (red) — Negative Response Codes
- **Busy** (orange) — 0x21 busyRepeatRequest responses

Custom Protocols

For vendor-specific protocols, see Plugins.

Plugins

Plugins extend CANcorder's protocol hint system to support vendor-specific diagnostic protocols.

Plugin Locations

Location	Purpose
App Store build: AppConfig/plugins	User plugins (sandboxed)
Direct download: `~/.config/cancorder/plugins/`	User plugins (JSON files)
Built-in	Bundled with application

App Store builds include a single example plugin (DynoTune Example Protocol) for reference.

Viewing Plugins

Open the **Plugins** menu to see:

- **Loaded plugins** — Green puzzle icon, click for details
- **Failed plugins** — Red X icon, click to see error

Plugin Details

Click a plugin to view:

- Title and description
- Plugin ID and version
- File path
- Services and subfunctions defined

Plugin Structure

Plugins are JSON files defining:

```
{
  "id": "my-protocol",
  "name": "My Protocol",
  "version": "1.0",
  "title": "My Custom Protocol",
  "subtitle": "For XYZ ECUs",
  "description": "Decodes proprietary diagnostic messages",
  "services": { ... }
}
```

Required Fields

Field	Description
`id`	Unique identifier
`name`	Internal name
`version`	Plugin version
`services`	Service definitions

Optional Fields

Field	Description
`title`	Display title in UI
`subtitle`	Shown below title
`description`	Full description
`carrierService`	UDS service used as wrapper
`prefixes`	Alternative message prefixes
`negativeResponseCodes`	Custom NRC meanings

Defining Services

```
{
  "services": {
    "62": {
      "name": "CustomRead",
      "subfunctions": {
        "48": {
          "name": "ReadVersion",
          "requestFields": [...],
          "responseFields": [...]
        }
      }
    }
  }
}
```

Service Properties

Property	Description
`name`	Service name for display
`subfunctions`	Map of subfunction ID → definition

Subfunction Properties

Property	Description
`name`	Subfunction name
`requestFields`	Fields in request message
`responseFields`	Fields in response message

Field Definitions

```
{
  "name": "Software Version",
  "offset": 1,
  "length": 2,
  "format": "version",
  "bigEndian": true
}
```

Field Properties

Property	Description	Default
`name`	Display name	Required
`offset`	Byte offset after SID	Required
`length`	Number of bytes	Required
`format`	Value format	`hex`
`bigEndian`	Byte order	`true`

Format Types

Format	Description	Example
`hex`	Hexadecimal	`4F0A`
`decimal`	Decimal number	`1024`
`ascii`	ASCII string	`ECU123`
`version`	Major.minor	`2.5`
`address`	Memory address	`0x80001234`

Carrier Services

Some protocols tunnel through standard UDS services. Use `carrierService` to specify the wrapper:

```
{
  "carrierService": 62,
  "services": {
    "62": {
      "subfunctions": {
        "48": { "name": "CustomCommand" }
      }
    }
  }
}
```

This tells CANcorder that service 0x3E (62) carries custom subfunctions.

Alternative Prefixes

If your protocol can be accessed via multiple UDS services:

```
{
  "prefixes": [
    {
      "bytes": [34, 255, 255],
      "description": "Via ReadDataByIdentifier 0xFFFF"
    }
  ]
}
```

Conversation Tracking

When a plugin claims a request, CANcorder remembers this for the ECU pair (TX ID ↔ RX ID). Subsequent responses from that ECU pair give the plugin first chance to decode.

This ensures consistent decoding within a diagnostic conversation.

Built-in Plugins

VehiCAL

Decodes the VehiCAL calibration protocol:

Subfunction	Name	Purpose
0x30	Ident	Check ECU patch status
0x31	ReadMemoryByAddress	Read memory (v1)
0x32	WriteMemoryByAddress	Write memory
0x33	ReadFastArray	Read via FastArray (v2)
0x35	ReadFastArrayInplace	Inline FastArray read

Creating Plugins

1. Create a JSON file in `~/.config/cancorder/plugins/`
2. Define your protocol structure
3. Restart CANcorder or reload

Example: Simple Protocol

```
{
  "id": "my-ecu",
  "name": "MyECU",
  "version": "1.0",
  "title": "My ECU Protocol",
  "services": {
    "34": {
      "name": "CustomService",
      "subfunctions": {
        "1": {
          "name": "GetStatus",
          "responseFields": [
            { "name": "Status", "offset": 0, "length": 1, "format": "hex" },
            { "name": "Counter", "offset": 1, "length": 2, "format": "decimal" }
          ]
        }
      }
    }
  }
}
```

Troubleshooting

Plugin Not Loading

Check the Plugins menu for failed plugins. Common issues:

- **JSON syntax error** — Validate your JSON
- **Missing required field** — Check id, name, version, services
- **Duplicate ID** — Plugin IDs must be unique

Hints Not Appearing

- Ensure ISO-TP Decode and Hints are enabled
- Check that your service ID matches the message
- Verify subfunction ID if used

J1939 Dashboard

The J1939 Dashboard provides a real-time overview of decoded J1939 parameter values with sparkline history charts, grouped by source ECU.

Opening the Dashboard

1. Set the transport mode to **J1939** in the Transport panel
2. Enable **Show Transport Messages**
3. Click the **dashboard icon** (grid) in the header toolbar

The dashboard replaces the split table view while active. Click the icon again to return to the tables.

How It Works

CANcorder decodes every J1939 payload message using the PGN catalog. Fields with numeric values (engine speed, temperatures, pressures, voltages, etc.) are tracked per source ECU and PGN.

Only fields whose value has **changed at least once** are shown. Static readings (e.g., a sensor that always reports the same value) are hidden to reduce clutter.

Dashboard Cards

Each card displays:

PGN name and label (e.g., EEC1 - Electronic Engine
Controller 1)
Field name (e.g., Engine Speed)
1234.5 rpm [sparkline chart ~~~~~~~~]
min max

- **Current value** shown in large text with color-coded units
- **Sparkline** shows the last 120 samples as a filled area chart
- **Min/max range** shown below the sparkline
- Values are sampled at most every 50 ms per field

Color Coding

Unit	Color
rpm	Orange
km/h	Green
deg C / °C	Cyan

Unit	Color
%	Yellow
V	Purple
kPa	Rose
km	Teal
(other)	Blue

ECU Groups

Cards are grouped by J1939 source address. Each group shows:

- ECU name from the J1939 address table (e.g., "Engine #1", "Transmission #1")
- Hex address (e.g., 0x00)
- Field count

Groups are sorted by source address. Within each group, fields are sorted by PGN and then field name.

PGN Catalog

The dashboard decodes fields from the built-in PGN catalog (EEC1, EEC2, ET1, CCVS1, VEP1, VD, EFL/P1) and any external catalog loaded from:

- `~/.config/cancorder/J1939db.json`
- `~/.config/pretty_j1939/J1939db.json`

See protocol-hints.md for details on the J1939 catalog format and loading.

Tips

- The dashboard processes messages incrementally, so switching to it after capturing data will catch up with all accumulated messages
- Clearing data (trash icon) resets the dashboard
- The dashboard works with both live captures and imported log files
- Light and dark themes are both supported

Quality Test

The Quality Test panel is meant for checking whether a CAN capture path is dropping, duplicating, or reordering frames.

What It Measures

- **Received**: Number of unique test frames received for the current run.
- **Missing**: Number of missing sequence numbers between the first and last received test frame.
- **Duplicates**: Frames with a sequence number that was already seen in the current run.
- **Reordered**: Frames that arrived after a newer sequence number had already been received.

Latency And Jitter

- **Relative Latency**: Extra receive delay compared with the fastest test frame in the current run. This is a relative metric, not an absolute end-to-end bus delay measurement.
- **Jitter**: Variation in packet spacing at the receiver. With a sender configured for `1 fps`, you should expect packet spacing near `1000 ms`; jitter shows how much that spacing varies.

Starting A New Test

Starting a new Quality Test clears the previous run's collected packets before measurement begins. If prior results are still visible, CANcorder asks for confirmation before discarding them.

Utilities

CANcorder has companion utilities for live scanning and capture that feed data into the app.

macOS

- **GS_USB + RusokuCAN**: Live scanning on macOS using GS_USB adapters and RusokuCAN.

Linux

- **SocketCAN**: Live scanning via SocketCAN adapters and tooling.

Links

Utilities are maintained here:

https://github.com/Automotive-Swift/CANcorder

Troubleshooting

Connection Issues

- Use the Connection menu to select a discovered logger (Zeroconf/mDNS)
- If no loggers appear, verify you're on the same network as the logger
- For manual connection, enter the logger's IP address and port
- Confirm the logger is reachable from your machine (same network or USB tether)
- If you are already connected, disconnect before loading example logs

No Frames Showing

- Check that the logger is sending frames (status indicator should be connected)
- Ensure filters are not hiding everything (reset hidden ECUs, clear ID filter)
- Toggle "Show Standard" and "Show Extended" to confirm addressing mode

J2534 Issues (Windows)

- **"Bridge executable not found"**: The J2534 bridge was not built. Run `make bridge` or `make dev` (which builds the bridge automatically).
- **Device shows but won't connect**: Check that the adapter is plugged in and powered. Some adapters need external 12V power.
- **Connected but no frames**: Verify the baud rate matches the CAN bus. CANcorder uses `CAN_ID_BOTH` for both 11-bit and 29-bit frame reception.
- **Bridge process died**: The J2534 DLL may have crashed. This is isolated from CANcorder — just reconnect. If it happens repeatedly, try a different driver version.
- **Diagnostic logging**: Set environment variable `J2534_BRIDGE_VERBOSE=1` to log all bridge requests to stderr.

ISO-TP Decode Looks Empty

- ISO-TP decoding depends on correct addressing (11-bit vs 29-bit)
- Some logs contain only raw CAN frames without ISO-TP traffic
- Turn on Hints to verify whether any messages look like UDS/OBD/KWP traffic

Plugin Issues

- Plugins load at startup from `~/.config/cancorder/plugins/*.json`
- Schema errors or invalid JSON will prevent loading; check the console logs
- If a plugin is too aggressive, restrict its matching rules (service IDs or subfunctions)

Performance Tips

- Very large logs can take time to parse and render; wait for parsing to complete

- For live capture at high frame rates, consider pausing auto-scroll or hiding ISO-TP
- Filtering with complex wildcards across millions of frames can be slow

Export Problems

- Export uses CL1000 format (Timestamp;Type;ID;Data)
- If you cannot save, verify the file permissions and destination path