

# edelkrone SDK

User Guide

# Contents

## 1 Getting Started

---

## 2 Connecting

---

### 2.1 edelkrone.connect()

---

### 2.2 Finding Units in Range: edelkrone.discover()

---

### 2.3 Layouts

---

### 2.4 Ending the Connection

---

## 3 The Device

---

### 3.1 Identity: device.info

---

### 3.2 State: device.status

---

## 4 Moving

---

### 4.1 Joystick Control: device.move.joystick()

---

### 4.2 Stopping

---

## 5 Poses

---

## 6 Focus

---

## 7 Settings

---

## 8 Tripod

---

## 9 Errors

---

## 10 Diagnostic Log

---

## 11 API Summary

---

# 1 Getting Started

The edelkrone SDK controls edelkrone motion control equipment from Python over Bluetooth Low Energy. It supports heads, sliders, jibs, dollies and tripods, used on their own or combined in a layout, and the focus motor fitted to a head. Release 0.1 covers connecting, reading the device's identity and state, joystick control, stored poses, focus, settings and the tripod.

The SDK requires Python 3.11 or newer; with an older Python, `pip` reports that no matching distribution was found. Install it with:

```
pip install edelkrone
```

|                                |                                                                                                                                                                                                                           |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Python</b>                  | 3.11 or newer.                                                                                                                                                                                                            |
| <b>Platforms</b>               | macOS, Windows and Linux. Bluetooth must be enabled. On macOS, the terminal or application requires Bluetooth permission (System Settings → Privacy & Security → Bluetooth); without it, no units are found.              |
| <b>Dependencies</b>            | <code>bleak</code> , the Bluetooth package, is installed automatically.                                                                                                                                                   |
| <b>Included in the package</b> | This guide, in <code>edelkrone/doc/</code> inside the installed package, and two example programs: <code>python -m edelkrone.examples.read_device</code> and <code>python -m edelkrone.examples.store_and_return</code> . |
| <b>Version</b>                 | <code>edelkrone.__version__</code>                                                                                                                                                                                        |

- Figure 1: Installation -

A first program:

```
import edelkrone

for unit in edelkrone.discover():
    print(unit.to_json())

with edelkrone.connect(name="HeadPLUS V3") as device:
    print(device.info.to_json())
    print(device.status.to_json())

    device.move.joystick(pan=0.3)
moving
    device.move.stop()

    device.poses.store(0)
    result = device.poses.move_to(0)
    if not result.is_success:
        print(result.reason)
```

The SDK follows three principles:

- **Calls are synchronous.** Each call returns when its operation has completed. No event loop is required. In asynchronous code, run a call in a worker thread, for example `await asyncio.to_thread(device.poses.move_to, 0)`.
- **The SDK polls the device.** After `connect()`, the SDK requests a new reading four times per second, and `device.status` always holds the latest one.
- **Refusals are returned; faults are raised.** A request the device declines, such as a move to an empty slot, returns a result that states the reason. A failure of the connection or of a command raises an exception (9).

**Not included in release 0.1.** Targets and point mode, driving to coordinates, creating a dolly path and reading the dolly position, and timelapse. These functions are planned for later releases.

## 2 Connecting

### 2.1 edelkrone.connect()

`connect()` finds the unit, connects to it, reads its identity and a first reading, joins its modules if required, starts polling and returns a `Device`. If any of these steps fails, it raises an exception that identifies the step (9).

```
device = edelkrone.connect(name="HeadPLUS V3", mac="A4:C1:38:5E:7B:2D", timeout=20.0)
```

| Argument                     | Default                     | Description                                                                                                                                                                                                                                                                                                     |
|------------------------------|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>name</code>            | None                        | The product name printed on the unit. Case and spaces are ignored: "HeadPLUS V3", "headplus v3" and "HEADPLUSV3" are equivalent.                                                                                                                                                                                |
| <code>mac</code>             | None                        | The unit's Bluetooth address, in any common notation: twelve hexadecimal digits, with <code>:</code> , <code>-</code> or <code>.</code> between them or none. Anything else is refused with <code>bad_argument</code> before the search. The address does not change when the unit is restarted.                |
| <code>with_modules</code>    | None                        | For units that have not been used together before: the addresses of the modules to join to this unit (2.3).                                                                                                                                                                                                     |
| <code>timeout</code>         | 20 s                        | How long to search for the unit before raising <code>not_found</code> ; a positive number of seconds. It limits the search only: connecting to a unit that has been found takes up to three attempts of 20 seconds each, three seconds apart.                                                                   |
| <code>transport</code>       | The desktop Bluetooth radio | A radio held by the program, from <code>edelkrone.open_radio()</code> , so that several calls share one radio. A radio passed here is not closed by <code>disconnect()</code> ; close it with <code>dispose()</code> . A radio opened by <code>connect()</code> itself is closed by <code>disconnect()</code> . |
| <code>config</code>          | Default values              | A <code>SessionConfig</code> that sets timeouts and the polling interval.                                                                                                                                                                                                                                       |
| <code>allow_narrowing</code> | False                       | Accepted, with no effect; a <code>DeprecationWarning</code> says so. A unit keeps the layout its modules form for as long as they are switched on (2.3).                                                                                                                                                        |

- Figure 2: connect() arguments -

Specify a name, an address, or both; when both are given, both must match. When neither is given, the SDK connects to the only unit in range. If more than one unit is in range, the call is refused and the message lists the units found. The address of a module that belongs to a layout is refused at once with `not_found`, and the message names the unit to connect to instead; so is the address of a module of a unit this program has open, which is driven through that unit's `Device`. A second `connect()` to a unit this program already has open, by its address or by its name, is refused with `already_connected`; to reach another unit of the same name, give its address.

To share one radio between several calls, open it with `open_radio()`:

```
radio = edelkrone.open_radio()
try:
    units = edelkrone.discover(transport=radio)
    with edelkrone.connect(mac=units[0].mac, transport=radio) as device:
        ...
finally:
    radio.dispose()
```

## 2.2 Finding Units in Range: `edelkrone.discover()`

`edelkrone.discover(seconds=8.0)` listens for the given time and returns every unit found as a list of `Seen`, ordered by signal strength. Opening and closing the radio add a few seconds to the call. It accepts the same `transport` argument as `connect()`. A unit this program has connected to is not listed, because a connected unit does not advertise, and neither are the modules of its layout. A module whose unit is connected elsewhere, for example to the edelkrone application, is listed on its own with the layout `group_member`; connect to its unit instead.

```
for unit in edelkrone.discover():
    print(unit.to_json())

device = edelkrone.connect(mac=edelkrone.discover()[0].mac)
```

```
{"name": "HeadPLUS V3", "mac": "A4:C1:38:5E:7B:2D", "signal": -51,
 "layout": "pan_tilt_and_slide", "modules": ["SliderPLUS V6"], "recognised": true}
```

| Field                                      | Description                                                                                                                                                                                    |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>name</code> , <code>mac</code>       | The product name and the Bluetooth address, as accepted by <code>connect()</code> .                                                                                                            |
| <code>signal</code>                        | The received signal strength in dBm, always below zero; values closer to zero indicate a stronger signal. The list is sorted by this field.                                                    |
| <code>layout</code> , <code>modules</code> | The layout the unit advertises and the product names of the modules in it. For a unit on its own, <code>layout</code> is <code>none</code> and <code>modules</code> is empty.                  |
| <code>recognised</code>                    | Whether the SDK supports this combination of units. An unrecognised combination is listed but cannot be connected to; so is a focus module, which works only through the head it is fitted to. |

- Figure 3: Seen -

**Connected units.** A unit connected to this program does not advertise and is not listed. A unit held by the edelkrone application on a phone may still be listed, but refuses the connection: `connect()` then raises `timed_out`. Close the other connection and try again.

## 2.3 Layouts

Modules mounted on a head, such as a slider, a jib or a dolly, are controlled through the head. The program connects to the head, which relays commands to its modules. `device.info.modules` lists the modules and `device.info.layout` names the layout, for example

`PAN_TILT_AND_SLIDE`.

The two three-piece products are also layouts, and in both the program connects to the head:

- **Krone X:** a HeadPLUS V3 with a JibPLUS and a DollyPLUS ( `PAN_TILT_DOLLY_AND_JIB_PLUS` ).
- **Krone S:** a HeadPLUS V3 with a SliderPLUS and a DollyPLUS ( `PAN_TILT_DOLLY_AND_SLIDER` ).

Units that have been used together before advertise their layout, and `connect()` requires no further arguments. Units that have not been used together must be joined once. Pass the module addresses in `with_modules`; the SDK joins them by cable or by radio, as the hardware requires:

```
device = edelkrone.connect(mac="A4:C1:38:5E:7B:2D", with_modules=["C8:F0:9E:11:22:33"])
```

A combination of units that the SDK does not support is refused with `NotSupported`, and the message names the units. If the unit does not take the modules named into its layout — for example a module that already belongs to another layout — `connect()` raises `layout_refused` rather than returning the unit in its old layout. The same holds when the unit takes the layout but its modules do not answer within 6 seconds: such a layout dissolves on its own. `connect()` returns when the modules answer the unit, so `status.modules_responding` and the module axes in `status.pose` are present from the start.

**A layout lasts while its modules are switched on.** The unit keeps the layout its modules form; it cannot be undone from the SDK. When a module is switched off, the device dissolves the layout — the head and its modules report `none` — and deletes the poses stored on the head; switching the module on again does not restore either. A refused move then names the change. Connect again with `with_modules` to form the layout again. Switching all units off and on returns each to its own.

**Before forming a different layout, switch all units off and on:** a module that was part of another layout since it was switched on does not join a new one, even when every unit advertises `none`. `connect()` then raises `layout_refused`, including when the new layout dissolves while the connection is being made.

**The positions of a Krone S.** In a Krone S the head reports the slider position from its own zero, which differs from the position the slider reports on its own. A pose stored on a Krone S includes the slider, and `move_to()` returns it with the head.

## 2.4 Ending the Connection

`device.disconnect()` stops polling and closes the connection. If `connect()` opened the radio, it closes the radio as well. Calling it more than once has no further effect. A `Device` is a context manager, so a `with` block closes the connection in all cases:

```
with edelkrone.connect(name="SliderPLUS V6") as device:  
    device.move.joystick(slide=0.2)  
    time.sleep(1.0)  
    device.move.stop()
```

**Stop the equipment before disconnecting.** Disconnecting does not stop moving equipment. Call `device.move.stop()` first, and place both calls in a `finally` block so that the equipment stops if the program fails.

### 3 The Device

`connect()` returns a `Device`. Its `info` and `status` describe the unit, and four groups provide the operations: `move`, `poses`, `focus` and `settings`. The `Device` itself manages the connection:

| Member                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>info</code>               | The unit's identity, read when connecting (3.1).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>status</code>             | The latest reading (3.2).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>status_updates</code>     | A stream that delivers each new reading; see <code>subscribe()</code> and <code>tap()</code> in 3.2.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>refresh()</code>          | Requests the identity, position and state from the device immediately.<br><br>Keeps <code>status.pose</code> up to date. The periodic reading reports the battery and the activity, but not the position; without position polling, <code>status.pose</code> holds the value from the last <code>refresh()</code> , except during joystick control, when the position is requested in any case. <code>poll_position()</code> requests the position every 250 ms, <code>poll_position(every=1.0)</code> once per second, and <code>poll_position(every=None)</code> stops. Disabled by default, because it doubles the radio traffic. |
| <code>is_connected</code>       | Whether the connection is currently open.<br><br>A stream of <code>True</code> and <code>False</code> values as the connection opens and closes, with the same <code>subscribe()</code> and <code>tap()</code> as <code>status_updates</code> . A lost connection is reported here at once; the SDK does not reconnect by itself.                                                                                                                                                                                                                                                                                                    |
| <code>connection_changes</code> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>disconnect()</code>       | Closes the connection. A <code>with</code> block calls it on exit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

- Figure 4: Device -

#### 3.1 Identity: `device.info`

An `Info` object, read when connecting and unchanged until the connection is closed. `to_dict()` returns it as a dictionary and `to_json()` as a JSON object; the field names match the Python attributes. `DeviceStatus`, `Seen` and `MoveOutcome` provide the same two methods; a `RigPose` is exported as part of the reading that contains it.

```
{
  "name": "HeadPLUS V3",
  "mac": "A4:C1:38:5E:7B:2D",
  "firmware": "0.31",
  "radio_firmware": "0.34",
  "firmware_ok": true,
  "layout": "pan_tilt_and_slide",
  "modules": ["SliderPLUS V6"],
  "axes": ["pan", "slide", "tilt"],
  "capabilities": ["focus", "manual_drive", "position", "settings", "stored_pose"]
}
```

| Field                                                                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>name</code> , <code>mac</code>                                           | The product name printed on the unit and its Bluetooth address.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>firmware</code> , <code>radio_firmware</code> , <code>firmware_ok</code> | The firmware versions of the device and of its radio module, and whether both meet the minimum versions this SDK supports. If <code>firmware_ok</code> is <code>false</code> , update the firmware with the edelkrone application before continuing.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>layout</code>                                                            | The modules the unit drives and their arrangement. <code>null</code> if the unit reports no layout. A head on its own reports <code>none</code> . A slider, jib or dolly on its own reports the one layout its model has ( <code>slide_only</code> , <code>jib_plus_only</code> , <code>dolly_only</code> and so on), while <code>discover()</code> shows the <code>none</code> the unit advertises.                                                                                                                                                                                                                                                                             |
| <code>modules</code>                                                           | The product names of the other units in the layout.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>axes</code>                                                              | The axes the layout can drive or read, as <code>RigAxis</code> values: <code>PAN</code> , <code>TILT</code> , <code>SLIDE</code> , <code>JIB_PAN</code> and <code>JIB_TILT</code> , shown in lower case in JSON. A head on its own has two axes; a head with a slider has three. A layout with a JibPLUS drives the arm through <code>jib_pan</code> and <code>jib_tilt</code> and has no <code>slide</code> ; a layout with a JibONE drives its boom through <code>slide</code> . <code>RigAxis</code> also has <code>FOCUS</code> , the axis of <code>RigPose.focus_cm</code> ; it is never listed here, because the focus motor is driven through <code>device.focus</code> . |
| <code>capabilities</code>                                                      | The functions the SDK supports on this model, as <code>Capability</code> values: <code>POSITION</code> , <code>STORED_POSE</code> , <code>MANUAL_DRIVE</code> , <code>FOCUS</code> , <code>SETTINGS</code> and <code>TRIPOD_LEVEL</code> , shown as <code>position</code> , <code>stored_pose</code> , <code>manual_drive</code> , <code>focus</code> , <code>settings</code> and <code>tripod_level</code> in JSON. A function not listed here is refused with <code>NotSupported</code> . To check before calling, use for example <code>Capability.FOCUS</code> in <code>device.info.capabilities</code> .                                                                    |

- Figure 5: Info -

## 3.2 State: `device.status`

A `DeviceStatus` object holding the latest reading, in degrees, centimetres, percent and seconds. It is held in memory, so reading it does not communicate with the device. Polling updates it four times per second, and `device.refresh()` requests a reading immediately. Any field may be `null` , which always means that the device has not reported the value: for example, a slider has no pan angle, a tripod reports no battery level, and before the first reading every field except `model` is `null` .

```
{
  "model": "HeadPLUS V3",
```

```

    "at": 1789822822.031,
    "activity": "idle",
    "pose": {"pan_deg": 89.96, "tilt_deg": 0.0, "slide_cm": 12.4, "focus_cm": null,
    "jib_pan_deg": null, "jib_tilt_deg": null},
    "slide_fraction": 0.31,
    "battery_percent": 82.0,
    "is_calibrated": true,
    "has_focus_motor": false,
    "layout": "pan_tilt_and_slide",
    "stored_pose_slots": [0, 3],
    "modules_responding": true,
    "move_progress": null,
    "position_is_trusted": true,
    "has_status_frame": true,
    "tilt_deg": 1.2,
    "is_level": true,
    "orientation_warning": false,
    "accel_x_ms2": null,
    "accel_y_ms2": null,
    "accel_z_ms2": null,
    "dolly_x_cm": null,
    "dolly_y_cm": null,
    "dolly_heading_deg": null
  }

```

| Field                                                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>model</code> , <code>at</code>                 | The product name of the unit that sent the reading, the same text as <code>info.name</code> , and the time the reading arrived as Unix time in seconds, as returned by <code>time.time()</code> ; <code>time.time() - status.at</code> is its age.                                                                                                                                                                                                                                                                           |
| <code>activity</code>                                | An <code>ActivityState</code> : <code>IDLE</code> , <code>MOVING</code> , <code>TRANSITIONING</code> or <code>TRACKING_TARGET</code> , shown as <code>idle</code> , <code>moving</code> , <code>transitioning</code> and <code>tracking_target</code> in JSON. The last two occur only while the edelkrone application is tracking a target with the head. <code>null</code> before the first reading. <code>is_moving</code> provides the same information as a Boolean.                                                    |
| <code>pose</code> , <code>position_is_trusted</code> | The position of the equipment as a <code>RigPose</code> : pan and tilt in degrees, the slider position in centimetres and the jib axes in degrees. Axes not present in the layout are <code>null</code> . The position always comes from a live reading; <code>position_is_trusted</code> is <code>true</code> once one has arrived. <code>focus_cm</code> is <code>null</code> unless the lens has been learned with the edelkrone application; the focus position as a percentage is <code>device.focus.percent()</code> . |
| <code>slide_fraction</code>                          | The slider or jib position as a fraction of its travel, from 0 to 1. Unlike the value in centimetres, it is included in every                                                                                                                                                                                                                                                                                                                                                                                                |

| Field                                                                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                    | reading. Reliable only once the slider has been referenced ( <code>is_calibrated</code> is <code>true</code> ); before that it may be <code>null</code> or a value that does not match <code>slide_cm</code> . A slider may run a movement of its own for a while after it is switched on, and reports <code>moving</code> meanwhile; wait for <code>idle</code> before moving it.                                                                                                                                                                                                                                                                                                      |
| <code>battery_percent</code>                                                       | The battery level. A tripod is powered from the mains and reports none.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>is_calibrated</code>                                                         | For a slider, whether it has been referenced; for a head, whether its level reference is set. <code>null</code> for models that require neither, and before the first reading. Information only: the SDK refuses no move on it, and a slider that is not referenced performs planned moves.                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>has_focus_motor</code>                                                       | Whether a focus motor is fitted, as reported by the device. <code>null</code> before the first reading.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>layout</code>                                                                | The layout most recently reported by the device.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>stored_pose_slots</code>                                                     | The slots that contain a stored pose.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>modules_responding</code>                                                    | Whether the modules in the layout respond to the unit. A module that advertises is powered, but not necessarily responding. <code>null</code> for a layout without modules.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>move_progress</code>                                                         | The progress of a planned move, from 0 to 1. Reported by heads; <code>null</code> on other units.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>has_status_frame</code>                                                      | Whether a complete reading has been received. Until then, the other fields are <code>null</code> . The property <code>is_known</code> returns the same value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>tilt_deg</code> , <code>is_level</code> , <code>orientation_warning</code>   | The unit's inclination from level in degrees, whether it is within 15°, and the device's orientation warning. Reported by PRO heads and by sliders; <code>null</code> on units without an inclination sensor.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>accel_x_ms2</code> , <code>accel_y_ms2</code> , <code>accel_z_ms2</code>     | The unit's accelerometer along its X, Y and Z axes, in m/s <sup>2</sup> . At rest the three make up gravity. Reported by heads, sliders, jibs, dollies and tripods whose firmware includes the accelerometer; <code>null</code> on older firmware and on a focus module. In a layout the reading is the head's; a module's own is read by connecting to it on its own. On a head the sensor is in its body, so the pan and tilt motors do not change it. The sensors are not calibrated: at rest the magnitude may differ from 9.81 m/s <sup>2</sup> by up to 0.5 m/s <sup>2</sup> . Read it at rest over a few seconds: just after a layout forms or a module moves it can read lower. |
| <code>dolly_x_cm</code> , <code>dolly_y_cm</code> , <code>dolly_heading_deg</code> | A dolly's place on the floor, X and Y in centimetres, and its heading — the direction it points — in degrees. Reported by                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

| Field | Description                                                                                                                                                                                         |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | a dolly whose firmware includes them, connected on its own; <code>null</code> on older firmware and on every other unit. In a layout the unit connected to is the head, which does not report them. |

- Figure 6: DeviceStatus -

To receive each reading as it arrives, subscribe to `status_updates` :

```
for status in device.status_updates.subscribe(max_pending=8):
    print(status.pose.pan_deg, status.battery_percent)
```

A subscription is a queue that the program reads at its own pace. `max_pending` sets how many readings it holds (256 by default), a whole number of 1 or more, or `None` for no limit; any other value is refused with `bad_argument` . When it is full, the oldest reading is discarded.

`next_event(timeout)` waits for one reading, and `cancel()` or a `with` block ends the subscription.

`tap(callback)` calls `callback(reading)` for each reading instead, on the SDK's delivery thread, and `untap(callback)` removes it. Keep a callback short and send no command from it: the reply could only be delivered on the thread that would be waiting for it, so the SDK refuses such a command with `CommandError` · `from_callback` . Hand the work to a thread of your own, or read `device.status` .

## 4 Moving

### 4.1 Joystick Control: `device.move.joystick()`

`joystick()` sets a speed for each axis, from  $-1$  to  $1$  as a fraction of the maximum speed. It does not set a target position. **The command must be repeated** approximately every 100 ms for as long as the equipment is to move; when commands stop arriving, the device goes on at the last speed for 1.5 seconds and then stops. Setting an axis back to zero stops it at once: a call in which every axis is zero, such as a stick returning to the centre, stops the axes that were moving. Call `stop()` to end the movement.

```
deadline = time.monotonic() + 2.0
try:
    while time.monotonic() < deadline:
        device.move.joystick(pan=0.3, tilt=-0.1)
        time.sleep(0.1)
finally:
    device.move.stop()
```

| Axis                                         | Movement                                                                                                                                                                                                                                                                                                                                   |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pan</code> , <code>tilt</code>         | The head.                                                                                                                                                                                                                                                                                                                                  |
| <code>slide</code>                           | The slider carriage. In a JibONE layout, the height of the jib arm.                                                                                                                                                                                                                                                                        |
| <code>jib_pan</code> , <code>jib_tilt</code> | The rotation and the tilt of the jib arm.                                                                                                                                                                                                                                                                                                  |
| <code>linear</code> , <code>angular</code>   | A dolly: forward and backward movement, and rotation in place. A dolly on its own moves freely; a dolly in a layout moves only along an existing path, created with the edelkrone application. Along a path only <code>linear</code> applies; on a free path, as the edelkrone application calls it, <code>angular</code> applies as well. |

- Figure 7: Joystick axes -

Axes not present in the layout ( `info.axes` ) are ignored, and a warning in the diagnostic log names the axis the first time; no other axis is moved in its place. `linear` and `angular` are not `RigAxis` values and never appear in `info.axes` : they apply whenever the layout has a dolly. Values outside  $\pm 1$  are limited to  $\pm 1$ . A value that is not a finite number is refused with `bad_argument` . If `stop()` is never called, the SDK treats the joystick as released 1.6 seconds after the last call.

### 4.2 Stopping

| Member                        | Description                                                                                                                                                                                                                                                                                                        |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>stop(settle=2.0)</code> | Stops all movement: joystick control, a move to a stored pose and the tripod column alike. It goes ahead of every pending command: the axes the joystick is driving are set to zero first, the planned move is aborted, and every other axis is set to zero. The call then waits up to <code>settle</code> seconds |

| Member                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                            | for the device to report that it is stationary and reads the position, so <code>status.pose</code> is where the equipment stopped; it typically returns within a second. The stop commands and the position read take time of their own, about 0.6 s on a unit on its own and about 1.3 s in a layout with modules, so <code>settle=0</code> skips only the wait for the device to report that it is stationary. It does not raise exceptions, so it can be used in a <code>finally</code> block: if the connection was lost during the movement, no second exception conceals the first. The one exception is a <code>settle</code> that is not a number of zero or more: the stop is sent without waiting, and then <code>bad_argument</code> is raised. |
| <code>is_moving</code>                     | Whether the latest reading reports the device as moving. <code>None</code> before the first reading, which differs from <code>False</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>wait_until_idle(timeout=60.0)</code> | Waits until the device reports that it is idle; returns <code>False</code> on timeout. <code>timeout</code> bounds the whole wait, and <code>0</code> asks once. After <code>joystick()</code> it waits until the equipment has come to rest, about 1.6 s after the last call. <code>poses.move_to()</code> already waits in this way.                                                                                                                                                                                                                                                                                                                                                                                                                     |

- Figure 8: device.move -

**The joystick brings the reading.** The device answers every joystick command with a reading, and `joystick()` returns when that answer has arrived; the SDK sends no request before the previous one has been answered. During joystick control the SDK therefore requests no readings of its own: `status_updates` delivers the device's answers, and the position is requested four to five times per second, so `status.pose` follows the movement. Read `device.status` in the joystick loop; `refresh()` is not needed there. Regular polling resumes approximately 250 ms after the last joystick command.

## 5 Poses

A pose is the position of every axis in the layout, stored in a numbered slot on the device.

`move_to()` returns the equipment to a stored pose; the device plans the motion so that all axes arrive at the same time.

```
device.poses.store(0)
result = device.poses.move_to(0, style=MoveStyle(speed=0.15, acceleration=0.4))
print(result.to_json())          # {"accepted": true, "completed": true, "duration": 3.4,
"reason": null}
```

| Member                                                                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>store(slot)</code>                                                      | Stores the current pose in <code>slot</code> and confirms from the next reading that the device has stored it; a slot that holds a pose is emptied first, so the confirmation is of the new pose. Returns <code>False</code> if the device did not store it, for example for a slot the firmware does not provide: a HeadPLUS V3 provides slots 0 to 11.                                                                                                                                                                                                                                  |
| <code>stored</code>                                                           | The slots that contain a pose, as last reported by the device. To test a slot: <code>slot in device.poses.stored</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>clear(slot)</code> , <code>clear_all()</code>                           | Deletes the pose in one slot, or in all slots.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>move_to(slot, style=None, start=None, timeout=60.0, verify=True)</code> | Moves to the stored pose and waits until the equipment is stationary. With <code>verify</code> enabled, the SDK reads the stored pose and the position the equipment reached, and reports a move as successful only if every axis of the layout arrived: within 0.25° on the head, 0.4° on a jib and 0.25 cm on a slider. A move in which the equipment did not move at all is reported as such. <code>start</code> names a slot to plan the move from, instead of the current position. A refusal is returned as a result, not raised. A loop ( <code>MoveStyle(loop=True)</code> ) does |

| Member                                                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>start_move_to(slot, style=None, start=None)</code> | <p>not finish and is refused here; start it with <code>start_move_to()</code>.</p> <p>Sends the move and returns immediately, for programs that control several devices and for loops: <code>start_move_to(1, start=0, style=MoveStyle(loop=True))</code> moves back and forth between the poses in slots 0 and 1 until <code>move.stop()</code>. <code>move.wait_until_idle()</code> waits for a move that ends. An empty slot is refused with <code>bad_argument</code>, since there is no result to carry the refusal.</p> |

- Figure 9: device.poses -

Slot numbers are whole numbers from 0 to 15 on a head and from 0 to 11 on a slider or a jib; any other value is refused with `bad_argument` before anything is sent. The protocol carries 16 slots on a head, but the firmware decides how many it stores: a HeadPLUS V3 stores 0 to 11, and `store()` returns `False` above that. A dolly and a tripod store no poses in slots, and every call of `device.poses` except `stored` raises `NotSupported` on them.

**A head moves from pose to pose.** When the head stands at a stored pose, or between two of them, a move to another pose runs through every slot in between, in order, as on the edelkrone application's keypose screen: from the pose in slot 2 to the pose in slot 0 it passes the pose in slot 1. Two conditions follow. An empty slot on the way stops the move, so the SDK refuses such a move with a reason that names the empty slots; two consecutive poses stored in the same place stop it there too. **Store poses in consecutive slots, each in a different place**, as the edelkrone application does. Away from every pose the move is direct.

The result of a move is a `MoveOutcome` :

| Field                   | Description                                                                                                                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>accepted</code>   | The device accepted the command.                                                                                                                                                                           |
| <code>completed</code>  | The device reported the move as finished and the equipment as stationary.                                                                                                                                  |
| <code>duration</code>   | The duration of the move in seconds.                                                                                                                                                                       |
| <code>reason</code>     | If the move did not take place, the reason, for example an empty slot, a layout that dissolved since connecting, equipment that did not move, or an axis that stopped short. <code>None</code> on success. |
| <code>is_success</code> | <code>True</code> when both <code>accepted</code> and <code>completed</code> are true.                                                                                                                     |

- Figure 10: MoveOutcome -

A `MoveStyle` defines how a move is performed. Its two values are fractions of the device's maximum: `speed` above 0 and at most 1, `acceleration` from 0 to 1. A value outside that range, or not a finite number, is refused with `bad_argument` when the move is sent.

```
MoveStyle(speed=0.15, acceleration=0.4)           # slow, with long acceleration and
deceleration
MoveStyle(speed=1.0, acceleration=0.01)           # the fastest possible move
MoveStyle()                                       # speed 0.5, acceleration 0.25: the
default of move_to()
MoveStyle(speed=0.3, acceleration=0.25, loop=True) # with start_move_to(slot, start=...):
back and forth until move.stop()
```

`speed` sets the speed. `acceleration` sets the share of the move used for acceleration at the start and deceleration at the end. `loop` repeats the move back and forth between the slot and `start`. For short and medium moves the duration depends mostly on `acceleration`; `speed` sets the top speed, which a short move may not reach.

**Short moves.** If the equipment is within approximately 3° of the stored pose, or a slider within approximately 1 cm, the device may not start the planned move; how short a move a unit still makes differs by device (a head was measured making 1.5° moves, a jib not starting 1.8°). When the device does not start and the equipment is away from the pose, `move_to()` returns a result with `is_success` set to `False` and a reason stating that the equipment did not move. Equipment already at the pose is a success. This is a property of the device and not a fault. A jib recalled together with a focus move may stop up to 0.35° short on its tilt.

**Axes stored in a pose.** A pose stores only the axes of the current layout. If a head is connected without its modules, its poses contain pan and tilt only. Check `device.info.layout` before storing poses.

## 6 Focus

A head with a focus motor sets focus as a **percentage of the lens travel**, from 0 at one end to 100 at the other. A focus module works only through the head it is fitted to; on its own it is refused at `connect()`. The motor must first learn the lens: it sets its reference, and then sweeps the lens once from one end to the other to find the travel. `setup()` performs these steps. The device keeps what it learned until it is switched off, so only the first `setup()` after switching on moves the lens, for 10 to 25 seconds depending on the lens; every later call returns at once. Keep the focus ring clear while it runs.

```
device.focus.setup()           # first, on every connection
reached = device.focus.move_to_percent(50)
print(device.focus.percent())  # current position of the focus ring
```

| Member                                               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>setup(timeout=120.0)</code>                    | Prepares the focus motor: the motor reference, then one sweep of the lens, each only if the device reports it missing. It must be called once on every connection, before anything else. Refused if no focus motor is fitted. The sweep is stopped after 45 seconds whatever <code>timeout</code> says, to protect the lens; if it found no end by then — no lens fitted, or a ring that turns freely — <code>setup()</code> raises <code>CommandError · focus_ends_not_found</code> . <code>timeout</code> limits the motor reference, and must be a positive number of seconds; any other value is refused with <code>bad_argument</code> . |
| <code>move_to_percent(percent, tolerance=1.0)</code> | Moves to a percentage of the travel, from 0 to 100, and returns the percentage the device reports when the ring has come to rest. It returns as soon as the ring is at rest within <code>tolerance</code> percentage points of the target, or wherever it came to rest. The device performs the movement at its own speed (about 35 seconds for the whole travel of a typical lens) and stops at the ends found by <code>setup()</code> .                                                                                                                                                                                                     |
| <code>percent()</code>                               | The current position of the focus ring, as the percentage of the travel the device reports.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

- Figure 11: device.focus -

**setup() is required before other operations.** On a unit that reports a focus motor, the following calls raise `CommandError · needs_focus_setup` until `setup()` has been called on the current connection: the joystick, all stored-pose operations, the settings, the tripod operations, `focus.move_to_percent()` and `focus.percent()`. The following calls are always available: all `reads(info, status, refresh(), poses.stored)`, `move.stop()` and `move.wait_until_idle()`, `disconnect()` and `setup()`.

The reason is that a stored pose on such a unit includes the focus position, which is measured on the travel `setup()` has the device learn. A pose stored or recalled before it would place the focus ring at an unintended position.

When `connect()` connects to a unit with a focus motor, it issues a `UserWarning` stating that `setup()` is required. To suppress it, use `warnings.filterwarnings("ignore", message=".*focus motor.*")`. Units without a focus motor are not affected.

Focus in centimetres, that is a distance to the subject, is not supported in this release; `status.pose.focus_cm` is `null` unless the lens has been learned with the edelkrone application.

## 7 Settings

| Member                         | Description                                                                                                                                                                                                                    |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>buzzer(muted=...)</code> | Mutes or unmutes the buzzer, and returns once the device has taken the setting: within 0.1 s, or about 0.7 s when settings are written one after another. <code>muted</code> must be <code>True</code> or <code>False</code> . |

- Figure 12: device.settings -

The device does not acknowledge settings. The call returns once the command has been sent.

## 8 Tripod

A motorized tripod levels itself and raises or lowers its column. The two operations are reached through `device.settings` and are refused on units other than tripods.

```
device.settings.level_tripod()           # starts the automatic levelling

end = time.monotonic() + 3.0             # raises the column for 3 seconds
while time.monotonic() < end:
    device.settings.drive_tripod(40)
    time.sleep(0.2)
device.settings.drive_tripod(0)          # stops the column
```

| Member                                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>level_tripod()</code>               | Starts the automatic levelling of the tripod.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>drive_tripod(relative_speed)</code> | Moves the column at a speed from <code>-100</code> to <code>100</code> : positive values extend it, negative values retract it, and <code>0</code> stops it. Values outside the range are limited to <code>-100</code> and <code>100</code> , a fraction is rounded to the nearest whole number, and a value that is not a finite number is refused with <code>bad_argument</code> . <b>It must be repeated</b> every 100 to 200 ms for as long as the column is to move, in the same way as the joystick; a single call moves the column for about 1.5 s, after which it stops by itself. <code>move.stop()</code> stops the column as well. |

- Figure 13: Tripod operations of `device.settings` -

The tripod does not acknowledge these commands. The calls return once the command has been sent. `status.activity` reads `moving` while the column moves and while the tripod levels, as the device reports it. After `drive_tripod(0)` it may go on reading `moving` for 1 to 1.5 s; `move.stop()` ends it at once. A tripod that is already level finishes levelling within a second. The tripod reports no inclination, so `tilt_deg` and `is_level` are `null`.

## 9 Errors

The SDK distinguishes two kinds of failure. A **refusal**, such as an empty slot or a move that did not start, is returned as a result with `is_success` and a `reason`. A **fault** raises one of three exceptions, all subclasses of `edelkrone.Error`. Each carries a `reason` code, a `message` and a `cause` with technical details for support requests.

```
try:
    device = edelkrone.connect(name="HeadPLUS V3")
except edelkrone.ConnectionError as e:
    print(f"{e.reason}: {e.message}")          # not_found: HeadPLUS V3 was not seen in 20.0
s. Seen: ...
```

| Exception and reason                                          | Cause                                                                                                                                                                                                                                                                                                                                                                      | Action                                                                                                                                                  |
|---------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ConnectionError</code> · <code>not_found</code>         | The unit was not found within the timeout: it is switched off, out of range, or connected to another program. Or the address given is that of a module in a layout, which is reached through its head; the message names the head.                                                                                                                                         | Check the power and close other connections to the unit. <code>discover()</code> lists the units in range.                                              |
| <code>ConnectionError</code> · <code>ambiguous</code>         | No name or address was given, and more than one unit is in range.                                                                                                                                                                                                                                                                                                          | Specify the unit. The message lists the units found and their addresses.                                                                                |
| <code>ConnectionError</code> · <code>no_radio</code>          | The computer has no Bluetooth adapter, or the Bluetooth package is not installed.                                                                                                                                                                                                                                                                                          | Check the adapter. Reinstall with <code>pip install edelkrone</code> .                                                                                  |
| <code>ConnectionError</code> · <code>no_permission</code>     | The operating system has not granted Bluetooth access to the program.                                                                                                                                                                                                                                                                                                      | macOS: System Settings → Privacy & Security → Bluetooth.                                                                                                |
| <code>ConnectionError</code> · <code>timed_out</code>         | The unit was found but did not accept the connection (three attempts of up to 20 seconds, three seconds apart), or did not respond to the first identity request. A unit held by the edelkrone application on a phone does this.                                                                                                                                           | Close any other connection to the unit; if none, restart the unit and reduce the distance.                                                              |
| <code>ConnectionError</code> · <code>dropped</code>           | The connection was lost while <code>connect()</code> was reading the identity or joining the modules. A connection lost later is reported by <code>connection_changes</code> , and commands then raise <code>not_connected</code> .                                                                                                                                        | Reconnect.                                                                                                                                              |
| <code>ConnectionError</code> · <code>already_connected</code> | The unit is already connected in this program.                                                                                                                                                                                                                                                                                                                             | Use the <code>Device</code> that <code>connect()</code> returned, or <code>disconnect()</code> it first.                                                |
| <code>ConnectionError</code> · <code>wrong_device</code>      | The unit connected but does not provide the services expected for its model, usually because of outdated firmware; or the unit reports a firmware error (its main firmware is not running, for example after an interrupted update), in which case <code>discover()</code> lists it with <code>recognised</code> set to <code>false</code> and no connection is attempted. | Check <code>info.firmware_ok</code> and update the firmware with the edelkrone application; after a firmware error, update it and switch it off and on. |

| Exception and reason                             | Cause                                                                                                                                                                                                                                           | Action                                                                                                                                                        |
|--------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ConnectionError · layout_refused</code>    | The unit did not take the modules named in <code>with_modules</code> into its layout, or took them in but they did not answer, for example because it already forms a layout with other modules, or a module belongs to another layout.         | Check that the modules are switched on. Switch all units off and on so that each advertises on its own, then name all modules in one <code>connect()</code> . |
| <code>CommandError · not_connected</code>        | A command was sent after the connection was lost.                                                                                                                                                                                               | Reconnect.                                                                                                                                                    |
| <code>CommandError · closed</code>               | A command was sent after <code>disconnect()</code> .                                                                                                                                                                                            | Connect again. A <code>Device</code> cannot be reused after it is closed.                                                                                     |
| <code>CommandError · no_reply</code>             | The device did not respond to a command in time.                                                                                                                                                                                                | Retry once. If the error persists, restart the unit.                                                                                                          |
| <code>CommandError · bad_reply</code>            | The device sent an unexpected response.                                                                                                                                                                                                         | Report the error with <code>info.firmware</code> and <code>cause</code> .                                                                                     |
| <code>CommandError · bad_argument</code>         | An argument is invalid, for example not a number, a slot out of range, a speed of <code>nan</code> or above 1, a timeout that is not a positive number of seconds, a loop given to <code>move_to()</code> , or a move started to an empty slot. | Correct the argument.                                                                                                                                         |
| <code>CommandError · from_callback</code>        | A command was sent from inside a <code>tap()</code> callback, where its reply could never be delivered.                                                                                                                                         | Hand the work to a thread of your own, or read <code>device.status</code> .                                                                                   |
| <code>CommandError · needs_focus_setup</code>    | The unit reports a focus motor and <code>setup()</code> has not been called on this connection, or the device has dropped what it learned. Only calls that change the equipment and the focus calls raise it.                                   | Call <code>device.focus.setup()</code> (6).                                                                                                                   |
| <code>CommandError · focus_ends_not_found</code> | <code>focus.setup()</code> swept the lens for 45 seconds without finding its ends, and stopped.                                                                                                                                                 | Check that a lens is fitted and the focus gear engages it.                                                                                                    |
| <code>NotSupported · unsupported</code>          | The model or layout does not provide the function, for example a pose on a tripod or a dolly, focus on a slider, or a focus module connected to on its own; or the units named in <code>with_modules</code> do not form a supported layout.     | Check <code>device.info.capabilities</code> .                                                                                                                 |

- Figure 14: Errors raised by the SDK -

**edelkrone.ConnectionError** is not Python's built-in `ConnectionError` . Refer to it as `edelkrone.ConnectionError`; `from edelkrone import *` replaces the built-in name in the importing module.

**Signal strength.** A unit at the edge of the radio range can be found by `discover()` but may fail after connecting, with `dropped` , `not_connected` or `no_reply` . Layouts with modules are affected first. If connections are unreliable, reduce the distance to the unit.

---

## 10 Diagnostic Log

The SDK writes diagnostic messages to `EdelkroneLog`. The log is disabled by default. To enable it, assign a function to `EdelkroneLog.sink` and, optionally, set `EdelkroneLog.minimum_level` to an `EdelkroneLogLevel`: `TRACE`, `DEBUG`, `INFO` (the default), `WARNING` or `ERROR`.

```
from edelkrone import EdelkroneLog, EdelkroneLogLevel

EdelkroneLog.sink = lambda level, message: print(f"[{level.name}] {message}")
EdelkroneLog.minimum_level = EdelkroneLogLevel.DEBUG
```

A sink is an `EdelkroneLogSink`: a function that takes the level and the message. It may be called from several threads, must not raise exceptions and must return quickly. To forward messages to Python's `logging` module, call a `logging` logger from the sink.

## 11 API Summary

The following table lists every name exported by `import edelkrone` and the section that describes it.

| Name                                                                                                                 | Description                                                                                                                                                                                | Section       |
|----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| <code>connect()</code> , <code>discover()</code> ,<br><code>open_radio()</code>                                      | Connect to a unit; list the units in range; open a radio to share between calls.                                                                                                           | 2             |
| <code>Device</code>                                                                                                  | A connected unit, with its identity, state and four operation groups.                                                                                                                      | 3             |
| <code>Info</code> , <code>DeviceStatus</code> , <code>Seen</code>                                                    | The identity, a reading, and a unit found in a scan. Each can be exported as JSON.                                                                                                         | 3.1, 3.2, 2.2 |
| <code>MoveControl</code> , <code>PoseController</code> ,<br><code>FocusControl</code> , <code>SettingsControl</code> | The types of <code>device.move</code> , <code>device.poses</code> , <code>device.focus</code> and <code>device.settings</code> , for use in type hints. They are not constructed directly. | 4, 5, 6, 7, 8 |
| <code>MoveOutcome</code> , <code>MoveStyle</code>                                                                    | The result of a pose move; the speed and acceleration of a move.                                                                                                                           | 5             |
| <code>ActivityState</code> , <code>Capability</code>                                                                 | The activity of a device; a function a model supports.                                                                                                                                     | 3.2, 3.1      |
| <code>RigPose</code> , <code>RigAxis</code> , <code>EdlSetup</code> ,<br><code>DeviceFamily</code>                   | A position, an axis, a layout, and a device family (head, slider, jib, dolly, focus module or tripod; a focus module is reached through its head).                                         | 3.2, 3.1, 2.3 |
| <code>SessionConfig</code> , <code>Transport</code>                                                                  | The types accepted by <code>connect(config=...)</code> and <code>connect(transport=...)</code> ; <code>open_radio()</code> returns a <code>Transport</code> .                              | 2.1           |
| <code>Error</code> , <code>ConnectionError</code> ,<br><code>CommandError</code> , <code>NotSupported</code>         | The base exception and the three fault types.                                                                                                                                              | 9             |
| <code>EdelkroneLog</code> , <code>EdelkroneLogLevel</code> ,<br><code>EdelkroneLogSink</code>                        | The diagnostic log, its levels and the sink type.                                                                                                                                          | 10            |
| <code>__version__</code>                                                                                             | The installed version.                                                                                                                                                                     | 1             |

- Figure 15: Exported names -

This content is subject to change.

If you have any questions about this document, please contact edelkrone.

Copyright © 2026 edelkrone.