Current API v5 — Software ≥ 4.1

An updated API client is available for EMScope software version **4.1 and later**. It extends the v4 interface with several protocol and usability improvements:

- **Binary amplitude frames**: measurement data is now transmitted as binary WebSocket frames (`[flags : uint8][padding : 3 bytes][float32 values …]`) instead of JSON, reducing overhead on high-frequency sweeps. The client decodes them transparently.
- **Cached frequency axis**: the server sends the frequency array only when the RBW or display range changes — not on every sweep. `self.freq` is updated automatically and shared across consecutive binary frames.
- **`has_sweep_data(msg)`**: a new helper method that returns `True` for both binary amplitude frames (EMI mode) and JSON `values` messages (impedance mode), replacing the manual `"values" in msg` check used in v4.
- **`repetition` flag**: `self.repetition` is set to `True` when a quasi-peak repeat sweep was triggered (200 Hz / 9 kHz bands only).
- **Convenience setters**: dedicated methods (`set_detector_type()`, `set_measure_channel()`, `set_display_range()`, `set_amp_units()`, etc.) alongside the generic `set_value()`.
- **Parallel multi-detector measurement**: an additional example demonstrates simultaneous PK + QP + AV acquisition using three concurrent WebSocket connections and `asyncio.gather`.
- **Reports with a verdict**: `request_report()` sends the frequency range and report fields the server needs (before, the report never arrived), and the reply sets `self.verdict` (`"PASS"`, `"FAIL"` or `"NOT_EVALUATED"`, which is never a pass), `self.detectors`, `self.not_evaluated`, `self.overload_traces` and `self.source` (the traces evaluated by a tabs report). Frequencies in `self.report` are in Hz. `set_value("trace_id", n)` names the trace in the first column of the report rows; without it the column is None.
- **Instrument in use**: if another session holds the instrument (another `session_uuid`), `connect()` raises `DeviceInUseError` instead of returning with empty device data.
- **More sweep flags**: `self.dwell_reduced` (the spectrogram fast mode shortened the dwell: quasi-peak and average uncalibrated) and `self.overload_held` (the trace contains data measured with overload).
- **User-defined modal channels**: `set_modal_coefficients()` and `set_neutral()` define the modal channels the same way as the advanced modal options of the web interface (see *Modal channels* below).
- **Number of points**: the spectrum has three interpolated points between bins (one in software 4.3 and earlier), about 23 475 points for the whole 9 kHz band. Take the length from `self.freq`; do not assume a fixed value.
- **Reference level**: the server starts at 115 dBuV (100 in software 4.3 and earlier). With the attenuator in `"auto"` it sets the attenuation, and with it the noise floor: call `set_reference_level()` at the start of every session.

The

api_v5.py

and

api_example_v5.py

files are provided. The

api_example_v5_multi_detector.py

file demonstrates simultaneous PK + QP + AV measurement using parallel connections. The

api_example_v5_modal.py

file measures user-defined modal channels on a three-phase instrument.

Required libraries: **websockets**, **matplotlib**. Python **3.8 or later** is required.

For a detailed description of the underlying WebSocket protocol, refer to the WebSocket API documentation.

Modal channels

In modal mode (`set_mode("modal")`) every modal channel is a weighted sum of the instrument inputs, computed before the FFT: `c1·In1 + c2·In2 + …`. The coefficients are defined with `set_modal_coefficients()` and require the **advanced_modal** license (`request_licenses()` lists it). Without the license the server ignores them and does not reply.

Configuration	Selected with	Modal channels	Inputs c1, c2, c3, c4
`emi`	RX2	cm, dm	L, N
`emi_3p_no_neutral`	RX4, `set_neutral(False)`	cm, dm1, dm2, dm3	L1, L2, L3

Configuration	Selected with	Modal channels	Inputs c1, c2, c3, c4
`emi_3p_neutral`	RX4, `set_neutral(True)`	`cm`, `dm1`, `dm2`, `dm3`	L1, L2, L3, N

- The argument is a dictionary `{configuration: {channel: {"c1": value, …}}}`. Each value is a real number or a `[real, imaginary]` pair, with both parts within -1 and +1 (+1 is applied as 0.99997). Only the configurations given are changed, but every channel given needs all the coefficients of its configuration; anything else raises `ValueError` before sending.
- The channel names are fixed; their content is free. For example, the common mode of L1 against the neutral, (L1 + N) / 2, is `cm` with c1 = 0.5, c2 = 0, c3 = 0, c4 = 0.5.
- The instrument measures two modal channels per sweep: `cm` with `dm1`, and `dm2` with `dm3`. With more than two traces the two pairs are measured in consecutive sweeps, so place the channels that must be measured at the same instant in the same pair.
- Every measuring connection must call `set_visible(True)`: on RX4 models the instrument measures only the inputs of the visible traces.
- With coefficients 1 and -1 a differential channel can reach twice the full scale of one input: it saturates and the trace shows OVERLOAD, as when an input overranges (before software 5.0: without an overload flag). The default channels use 0.5 and -0.5: add 6.02 dB to obtain the full difference.
- The coefficients apply to the RBWs up to 30 MHz. With RBW `"120"` (30 - 110 MHz) the modal channels are fixed.
- Send them at the start of every session: a server restart returns them to the defaults. Software 4.3 and earlier also lose them on a band change that loads another firmware (for example to `"120"` and back); there, send them again after `set_rbw()`.

The

api_example_v5_modal.py

file measures the common and the differential mode of each phase against the neutral (six channels) with two coefficient configurations, one after the other.

Legacy APIs

Software ≤ 3.13

The

api.py

and

api_example.py

files are provided for use with EMScope software versions **3.13 and earlier**.

Required libraries: **websockets, json**. The **matplotlib** library is used in the example script.

API v4 — Software = 4.0

A fully redesigned API client is available for EMScope software version **4.0**. It introduces a cleaner interface, automatic RBW and display range management, robust session handling, and transparent keepalive ping/pong support.

The

api_v4.py

and

api_example_v4.py

files are provided.

Required libraries: **websockets**, **matplotlib**. Python **3.8 or later** is required.

From:
https://emzer.com/wiki/ -

Permanent link:
https://emzer.com/wiki/doku.php?id=public:emscope:python_api

Last update: **2026/11/04 17:53**