

# GUI Backend User Guide

---

## Audience and Scope

- Written for payload operators, ground-station technicians, and anyone integrating a dashboard with the GUI backend.
- Focuses on day-to-day control, telemetry consumption, and troubleshooting. For internal APIs or implementation notes, see [src/gui\\_backend\\_developer\\_guide.md](#).

## System Overview

- The backend exposes a newline-delimited ASCII control surface on TCP port **1029**. Up to eight clients can stay connected at the same time.
- Every open socket receives an unsolicited JSON telemetry snapshot once per second. Direct commands produce framed replies.
- Cached state covers station metadata, RF configuration, antenna position, satellite telemetry, pass predictions, and a ring buffer of recent events.
- Command handling is case-insensitive; responses always begin with **OK** or **ERROR**.

## Prerequisites and Quick Checklist

### Access Requirements

- IP reachability to the host running **gui\_backend\_start()** (usually the ground-station controller).
- Firewall rule permitting outbound TCP 1029 from your workstation or automation host.
- (Optional) SSH access to the controller if you need to restart the backend.

### Suggested Tools

- **nc**, **ncat**, **socat**, or another TCP terminal for manual sessions.
- A scripting environment (Python, Node, etc.) if you plan to parse telemetry or automate command sequences.
- JSON-aware viewer to inspect the periodic snapshots.

### System Pre-requisites

- Primary verification platform: Ubuntu 16.04 LTS (4.4 series kernel) on an x86\_64 workstation with 8 GB RAM.
- Integrated hardware-in-the-loop setup: production ground-station controller running the RF front-end, rotator interface, and **gui\_backend\_start()** service.
- Python toolchain: Ubuntu 16.04 ships Python 2.7.12 and Python 3.5.2; the Waf-based build scripts and optional CSP Python bindings operate correctly with CPython  $\geq 2.5$  ([lib/libcsp/src/arch/windows/README](#)).
- Ensure compatible serial adapters and CSP radio hardware are available if you plan to issue RF or rotator commands from the user guide steps.

### Pre-Session Checklist

1. Confirm the backend process is running (check **ps** or system supervisor logs).
2. Ensure no more than eight users are already connected; otherwise connections will be refused.
3. Decide whether you are only monitoring (read-only) or also issuing state-changing commands; coordinate with other operators accordingly.

## Connecting and Verifying the Link

1. Open a TCP session to port **1029**.
2. Immediately send **PING** to confirm the socket is responsive.
3. Issue **STATUS** to pull a full snapshot framed with **OK STATUS / END**.
4. Keep the socket open to continue receiving unsolicited **"type":"telemetry"** JSON once per second.

## Sample Interactive Session

Terminal transcripts in this guide use diff-colored blocks:

lines starting with **+** **\$** represent user input (green) and **-** **>>** represent backend responses (red).

```
+ $ nc groundstation.local 1029
+ $ PING
- >> OK PONG
+ $ STATUS
- >> OK STATUS
- >> {"type":"status","station":{"name":"GS-
ALPHA","mode":"IDLE","emergency_stop":false,"lat":1.2976,"lon":103.7803,"a
lt_m":40.2,"true_north_deg":2.0,"time_utc":"2025-10-
20T03:41:05Z","time_local":"2025-10-20T11:41:05"},"antenna":
{"az_deg":45,"el_deg":10,"last_command_success":true},"rf":
{"tx_hz":437505000,"rx_hz":145950000},"satellite":
{"norad":99555,"lat_deg":-12.4,"lon_deg":110.3,"alt_km":520.1,"velocity_km
_s":7.5,"range_km":1320.4,"range_rate_km_s":-1.2,"tle_age_sec":86400},"pas
ses":[{"name":"CSP-11","aos":"2025-10-20T03:55:12Z","los":"2025-10-
20T04:05:55Z","duration_sec":643,"peak_elevation_deg":62}],"faults":[]}}
- >> END
```

Notes:

- Terminate each command with **\n**. **\r\n** is also accepted; trailing **\r** characters are stripped.
- If you close the socket, reconnect to keep receiving telemetry. There is no reconnection backoff from the server.

## Working With the Telemetry Stream

- **STATUS** replies include **OK STATUS** + JSON + **END** framing and are returned only on demand.
- The periodic broadcast omits the framing, sets **"type":"telemetry"**, and arrives at ~1 Hz per connected client.
- Snapshot generation is atomic; each JSON document contains a consistent view of station, antenna, RF, satellite, pass, and event-derived data.

- Empty fields are still emitted (e.g., `faults: []`) so client parsers should tolerate default values.

JSON Section	Key Fields	Interpretation
<code>station</code>	<code>name</code> , <code>mode</code> ( <code>IDLE</code> , <code>TRACKING</code> , <code>MAINTENANCE</code> ), <code>emergency_stop</code> , <code>lat/lon/alt</code> , <code>true_north_deg</code> , UTC/local timestamps	Identifies the ground station and surfaced safety state.
<code>antenna</code>	<code>az_deg</code> , <code>el_deg</code> , <code>last_command_success</code>	Latest commanded position and whether the last rotator action succeeded.
<code>rf</code>	<code>tx_hz</code> , <code>rx_hz</code>	Doppler-adjusted uplink/downlink center frequencies in Hz.
<code>satellite</code>	<code>norad</code> , <code>lat_deg</code> , <code>lon_deg</code> , <code>alt_km</code> , <code>velocity_km_s</code> , <code>range_km</code> , <code>range_rate_km_s</code> , <code>tle_age_sec</code>	Most recent orbital snapshot feeding pointing and pass predictions.
<code>passes[]</code>	<code>name</code> , <code>aos</code> , <code>los</code> , <code>duration_sec</code> , <code>peak_elevation_deg</code>	Up to 16 scheduled passes; timestamps are ISO-8601 UTC.
<code>faults[]</code>	Reserved	Currently always empty; future releases may populate it with latched alarms.

Consuming the Stream

- Prefer long-lived connections to avoid missing telemetry. If you must reconnect, discard partial JSON (no chunked framing).
- Treat the `"type"` field as the discriminator when multiplexing command responses and telemetry.
- When writing custom clients, add a read loop that splits on `\n`, buffers until JSON braces balance, then parse.

Command Reference

General Commands

Command	Description	Example & Response
<code>PING</code>	Liveness probe.	<code>PING</code> → <code>OK PONG</code>
<code>HELP</code>	Lists supported commands.	<code>HELP</code> → text list ending with <code>END</code>
<code>STATUS</code>	Returns a framed snapshot immediately.	<code>STATUS</code> → <code>OK STATUS</code> , JSON, <code>END</code>

Station Mode and Safety

Command	Description	Usage Notes
---------	-------------	-------------

Command	Description	Usage Notes
<b>SET_MODE</b> <idle\ tracking\ maintenance>	Changes the station operating mode.	Returns <b>OK SET_MODE &lt;value&gt;</b> ; actual modes are case-insensitive ( <b>maint</b> also accepted). Verify via telemetry.
<b>SET_EMERGENCY</b> <true\ false\ 1\ 0\ on\ off>	Engages or clears the emergency-stop latch.	Replies with <b>OK SET_EMERGENCY true false</b> and logs an event (severity <b>ERROR</b> when engaging).

## RF and Pointing

Command	Description	Usage Notes
<b>SET_SAT</b> <1-255>	Selects the active satellite ID (calls <b>mcs_sat_sel</b> ).	On success: <b>OK SATELLITE</b> . Invalid IDs return <b>ERROR Invalid satellite id</b> .
<b>SET_TX</b> <freq_hz>	Sets the uplink carrier frequency.	Range: unsigned 32-bit. Success yields <b>OK TX &lt;hz&gt;</b> ; failure returns <b>ERROR Frequency configuration failed</b> .
<b>SET_RX</b> <freq_hz>	Sets the downlink carrier frequency.	Same parsing rules as <b>SET_TX</b> .
<b>SET_AZEL</b> <az_deg> <el_deg>	Sends a rotator pointing command via <b>serial_set_az_el</b> .	Valid ranges: az -360..360, el -90..180. The handler does <b>not</b> emit an <b>OK</b> reply; rely on telemetry or rotator events to confirm motion.

## Packet and History Commands

Command	Description	Usage Notes
<b>SEND_PACKET</b> <pri> <src> <dst> <dst_port> <src_port> <hmac> <xtea> <rdp> <crc> <hex_payload>	Sends a CSP packet via <b>send_packet_struct</b> .	<b>pri</b> 0..3, ports 0..63, security flags 0/1, payload must be even-length hex. Replies <b>OK SEND_PACKET &lt;bytes&gt;</b> on success.
<b>LAST_UPLINK</b>	Summarizes the most recent uplink transaction.	<b>OK LAST_UPLINK</b> origin=... bytes=... status=success failure file=/path.
<b>LAST_DOWNLINK</b>	Shows the most recent downlink (source/destination nodes, bytes, file path).	<b>OK LAST_DOWNLINK</b> origin=... bytes=... src=X dst=Y file=....

Command	Description	Usage Notes
<code>GET_EVENTS [count]</code>	Dumps the newest telemetry/events ring buffer entries (default 64).	Starts with <code>OK EVENTS &lt;n&gt;</code> , emits timestamped lines, ends with <code>END</code> .

## Response Conventions

- Any parsing or validation issue returns `ERROR <reason>`.
- All numeric arguments are parsed as base-10 by default; a `0x` prefix enables hexadecimal.
- Commands are atomic; you do not need to wait between requests, but avoid flooding (stick to <10 commands/sec) to keep buffers manageable.

## Operational Playbooks

### Monitor an Upcoming Pass

- Connect and issue `STATUS` to prime the cache.
- Watch the `passes` array; each entry includes `aos`, `los`, and `peak_elevation_deg`.
- While tracking, observe `antenna.az_deg/el_deg` and `satellite.range_km` to ensure values evolve smoothly.
- Use `GET_EVENTS 10` for a concise history of uplink/downlink attempts during the pass.

### Change Station Mode

- Announce the intent to other operators (avoid conflicting commands).
- `SET_MODE tracking` (or `maintenance` / `idle`).
- Confirm the response `OK SET_MODE ...`, then verify `station.mode` flips in the next telemetry JSON and an INFO event is logged.

### Handle an Emergency Stop

- To engage: `SET_EMERGENCY true`. All downstream software should treat `station.emergency_stop=true` as a hard inhibit.
- Investigate and clear the root cause.
- To release: `SET_EMERGENCY false`. Verify the flag changes and that new rotator/RF commands succeed.

### Tune RF Chains

- Consult the mission plan or Doppler prediction to determine the required center frequencies.
- Run `SET_TX <hz>` and/or `SET_RX <hz>`; expect `OK TX ... / OK RX ...`.
- Confirm telemetry reflects the updated `rf.tx_hz/rf.rx_hz`.

### Point the Antenna

- Ensure emergency stop is clear and rotator hardware is ready.
- Issue `SET_AZEL <az> <el>`.

- 3. Because the handler does not return an **OK**, immediately monitor telemetry and **GET\_EVENTS** for a **"Rotator command"** entry. The event detail includes the commanded az/el and whether it succeeded.

Send a CSP Packet

- 1. Encode the payload as hex (even number of characters).
- 2. Issue **SEND\_PACKET ...** with the correct header fields (ports 0–63).
- 3. On **OK SEND\_PACKET <len>**, watch the telemetry/events feed for an **UPLINK** entry confirming the result.

Event History and Auditing

- The backend stores 64 events in a ring buffer; **GET\_EVENTS** paginates from newest to oldest.
- Each line follows **YYYY-MM-DDTHH:MM:SSZ <SEVERITY> <ORIGIN> | <SUMMARY> (<DETAIL>)**.
- Severity mapping: **UPLINK**, **DOWNLINK**, **INFO**, **ERROR**.
- Use **GET\_EVENTS 5** for quick spot checks, or omit the argument to dump the entire buffer.

Sample Output

```
+ $ GET_EVENTS 3
- >> OK EVENTS 3
- >> 2025-10-20T03:40:55Z INFO GUI | Station mode (TRACKING)
- >> 2025-10-20T03:40:57Z UPLINK GUI | Uplink transmission (success
bytes=2048)
- >> 2025-10-20T03:41:02Z INFO ROTATOR | Rotator command (az=45 el=10
success)
- >> END
```

Troubleshooting

Symptom	Likely Cause	Resolution
Connection refused	Backend not running or already has eight clients.	Restart the process or ask another user to disconnect.
<b>ERROR</b> <b>Unknown command</b>	Typo or unsupported instruction.	Run <b>HELP</b> to verify spelling; commands are case-insensitive but arguments are positional.
<b>ERROR</b> <b>Invalid ... responses</b>	Out-of-range numeric argument or malformed hex payload.	Double-check bounds listed in this guide; <b>SEND_PACKET</b> payload must be even-length hex.
No reply to <b>SET_AZEL</b>	Handler intentionally silent.	Confirm via telemetry ( <b>antenna.last_command_success</b> ) or look for rotator events.

Symptom	Likely Cause	Resolution
Telemetry stops updating	TCP socket dropped or backend hung.	Reconnect; if issue persists, inspect backend logs and consider restarting <code>gui_backend_start()</code> .
Frequent disconnects	Client not reading fast enough or network idle timeout.	Ensure your script drains the socket continuously; disable TCP keepalive timeouts if necessary.

Quick Reference

- Port: `1029/TCP`. Protocol: ASCII commands + JSON telemetry.
- Command cadence: keep under 10 commands/sec per client to avoid buffer churn.
- Always verify state changes through telemetry or `GET_EVENTS`; do not rely solely on immediate command replies.
- When scripting, treat any line starting with `{` as JSON and everything else as textual status.
- Remember to release emergency stop and confirm station mode before scheduling autonomous passes.

v1.0 – 12 NOV 2025 v1.1 - 14 NOV 2025