

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 >= 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, "alt_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}, "passes": [{"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
station	<code>name, mode (IDLE, TRACKING, MAINTENANCE), emergency_stop, lat/lon/alt, true_north_deg, UTC/local timestamps</code>	Identifies the ground station and surfaced safety state.
antenna	<code>az_deg, el_deg, last_command_success</code>	Latest commanded position and whether the last rotator action succeeded.
rf	<code>tx_hz, rx_hz</code>	Doppler-adjusted uplink/downlink center frequencies in Hz.
satellite	<code>norad, lat_deg, lon_deg, alt_km, velocity_km_s, range_km, range_rate_km_s, tle_age_sec</code>	Most recent orbital snapshot feeding pointing and pass predictions.
passes[]	<code>name, aos, los, duration_sec, 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
PING	Liveness probe.	<code>PING → OK PONG</code>
HELP	Lists supported commands.	<code>HELP → text list ending with END</code>
STATUS	Returns a framed snapshot immediately.	<code>STATUS → OK STATUS, JSON, END</code>

Station Mode and Safety

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

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

RF and Pointing

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

Packet and History Commands

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

Command	Description	Usage Notes
<code>GET_EVENTS [count]</code>	Dumps the newest telemetry/events ring buffer entries (default 64).	Starts with <code>OK EVENTS <n></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

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

Change Station Mode

1. Announce the intent to other operators (avoid conflicting commands).
2. `SET_MODE tracking` (or `maintenance / idle`).
3. 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

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

Tune RF Chains

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

Point the Antenna

1. Ensure emergency stop is clear and rotator hardware is ready.
2. 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**, **DLINK**, **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.
ERROR Unknown command	Typo or unsupported instruction.	Run HELP to verify spelling; commands are case-insensitive but arguments are positional.
ERROR Invalid ... responses	Out-of-range numeric argument or malformed hex payload.	Double-check bounds listed in this guide; SEND_PACKET payload must be even-length hex.
No reply to SET_AZEL	Handler intentionally silent.	Confirm via telemetry (antenna.last_command_success) 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