HoloQ Player - Configuration Specification (config.json)

HoloQ Technologies

2026

Contents

HoloQ Player — Configuration Specification (config.json)

Version: 2.4 Format: JSON (UTF-8). `//` and `/* */` comments and trailing commas are tolerated. Consumed by: HoloQ Player (Android).

---

1. Purpose

`config.json` holds device/deployment settings: serial link, playback buffering, audio output, storage selection, the playlist location, logging, heartbeat/stall behavior, and the serial command byte map (reboot, transport/speed). It is read once at startup. **Every field is optional** — an empty `{}` yields a fully working default configuration (serial ON at `/dev/ttyAS5` 9600 8N1, analog audio, removable storage).

Location: the app searches, in order — app dir (`<externalFiles>/holoq/config.json`), `/sdcard/holoq/config.json`, `/sdcard/config.json`, `<removable card>/config.json`, `<removable card>/holoq/config.json` — and uses the first readable file.

---

2. Top-Level Structure

```
{
  "serial": { ... },
  "buffer": { ... },

  "audio_output": "hdmi",
  "audio_format": "s16",
  "volume": 100,

  "hwdec": true,

  "storage": "removable",
  "playlist_path": "playlist.hlq",
```

```
  "debug_log": false,
  "log_file": "holoq.log",
  "log_max_mb": 10,
  "log_rotations": 3,

  "watchdog_heartbeat_send": 247,
  "watchdog_interval_ms": 5000,
  "stall_detect_ms": 5000,

  "reboot_receive": 255,

  "stop_receive": 254,
  "play_right_receive": 253,
  "play_left_receive": 252,
  "speed_fast_receive": 251,
  "speed_regular_receive": 250,
  "speed_slow_receive": 249,
  "speed_ultraslow_receive": 248
}
```

---

3. Fields

3.1 `serial` (object)

Field	Type	Default	Description
enabled	bool	true	Serial link on/off. Serial is ON by default; set false to disable.
port	string	“uart5”	Symbolic `uart0`..`uart5` (→ `/dev/ttyAS<n>`) or an explicit `/dev/...` path.
mode	string	“binary”	`"binary"` or `"ascii"`.
baud	int	9600	Baud rate; must match the PC.
data_bits	int	8	5–8.
parity	string	“none”	`"none"` | `"even"` | `"odd"`.
stop_bits	int	1	1 or 2.
invert	bool	false	Logic inversion (if the link requires it).
handshake	bool	false	Require the start-signature handshake before acting on any received byte. **Off by default** — see rationale below for when to turn it on.
start_signature	string	“HOLOQSTART”	Printable ASCII (0x20–0x7E), max 32 characters. Non-conforming characters are stripped; if nothing usable remains, the default is substituted.
handshake_ack_send	int	246	Byte sent once, the moment the start signature is matched. Outside range 0–255 suppresses the ack.
signature_gap_ms	int	2000	Max gap allowed between consecutive signature bytes. Clamped 0–60000; `0` disables the timing check.

Back-compat: if there is no `serial` object, legacy flat keys `serial_device` (device path) and `serial_baud` are used, defaulting to `/dev/ttyAS5` 9600 8N1. Serial defaults ON so a missing/partial config never silently kills the control link.

The UART is root-owned; the app configures it via `su` (`stty` raw mode), reads with a root `cat`, writes via a persistent root writer. **Root is required for serial.**

Why the handshake exists, and why it's off by default: the serial link carries no framing, addressing, or checksum, so without some way to distinguish a deliberate command from whatever else is on the wire — a floating RX line, a USB-serial adapter's power-up break, a controller bootloader's own startup traffic — any of those can coincidentally match a valid command byte. This matters for deployments with an external controller (a PC, Arduino, Raspberry Pi, etc.) that need protection against spurious commands during power-up. It is irrelevant for the many deployments that run with no controller at all — straightforward auto-play/looping signage — so `handshake` defaults to `false` and the Player behaves exactly as it always has: every received byte is acted on immediately, no signature required. **Set `handshake: true` if you're integrating an external controller and want this protection** — once enabled, the Player ignores every received byte, `reboot_receive` included, until the controller sends `start_signature` and receives `handshake_ack_send` back. Full behavior, timing, and the controller-side implementation pattern are covered in the Serial Protocol Integration Guide §3.1 — this table is the authoritative field reference; that guide is the authoritative behavioral spec.

3.2 `buffer` (object) — ExoPlayer read-ahead

Field	Type	Default	Description
min_ms	int	15000	`minBufferMs` — keep at least this buffered.
max_ms	int	60000	`maxBufferMs` — buffer up to this much.
playback_ms	int	1000	`bufferForPlaybackMs` — start playing after this.
playback_after_rebuffer_ms	int	2000	`bufferForPlaybackAfterRebufferMs`.

Values are clamped so the DefaultLoadControl invariants hold (max≥min, min≥playback, min≥after-rebuffer). Larger min/max = smoother on slow SD cards, at the cost of RAM and a longer initial fill.

3.3 Audio

Field	Type	Default	Description
audio_output	string	"hdmi"	Which physical output(s) carry audio: `"analog"` | `"hdmi"` | `"both"`. See below.
audio_format	string	"s16"	PCM format hint (informational on Android).
volume	int	100	Global default 0–100; per-item `volume` overrides.

How `audio_output` works (P6-012). On this board the vendor audio HAL decides which outputs are live from the vendor property **`vendor.audio.output.active`**, whose factory value is `AUDIO_CODEC,AUDIO_HDMI` — i.e. the HAL opens BOTH the analog PCM (card0 `audiocodec`) and the HDMI PCM (card1 `ahubhdmi`), which is why sound appeared on HDMI *and* analog. It is **not** a kernel/AHUB mirror. At startup — before any audio plays — the app sets this property to match `audio_output` and restarts the vendor audio HAL (`ctl.restart vendor.audio-hal`):

`audio_output`	`vendor.audio.output.active`	Result
`analog`	`AUDIO_CODEC`	HDMI silent, analog only
`hdmi`	`AUDIO_HDMI`	analog silent, HDMI only **(default)**
`both`	`AUDIO_CODEC,AUDIO_HDMI`	factory behaviour: sound on both

Notes: - **Requires root (su)** — the same dependency the serial link already has. - The property is **not** a `persist.*` one, so it reverts to the factory value on every boot; the app therefore re-applies it at each

startup. - If the property already matches, the HAL is **not** restarted (no needless delay). - After a restart the app waits ~1.5 s for the HAL to come back before starting playback. - An unrecognised value leaves HAL routing untouched (logged as a warning). - `audio_output` also sets ExoPlayer's *preferred* device within the outputs the HAL has enabled; `"both"` expresses no preference. - **Removed in P6-012:** the old `hdmi_pcm_node` key. Blocking the HDMI PCM node silenced analog too (the HAL's HDMI write failed and it tore the whole output down), so that approach is gone.

3.4 Video

Field	Type	Default	Description
hwdec	bool	true	Require hardware decode (cedar VPU). Keep true in prod.

3.5 Storage & playlist

Field	Type	Default	Description
storage	string	"removable"	`app` → app external files; `internal/sdcard` → `/storage/emulated/0`; `removable/sdcard_ext` → the mounted removable card (discovered at runtime, no UUID in config).
playlist_path	string	"playlist.hlq"	Playlist path; relative resolves under the storage base. (CR-02: was "playlist.json".)

3.6 Logging

Field	Type	Default	Description
debug_log	bool	false	Verbose logging.
log_file	string	none	Log file name/path (under storage if relative).
log_max_mb	int	10	Max size per log file before rotation.
log_rotations	int	3	Rotated files to keep.

3.7 Liveness / watchdog

Field	Type	Default	Description
watchdog_heartbeat_send	int	247	Heartbeat byte sent to the PC.
watchdog_interval_ms	int	5000	Heartbeat period (ms).
stall_detect_ms	int	5000	No playback progress for this long → STOP the heartbeat, no other action.

Disabling the heartbeat entirely: `watchdog_heartbeat_send` is parsed as a plain integer with no range coercion at parse time (unlike e.g. `volume`, which is clamped 0–100). The Player only actually sends the heartbeat byte if `watchdog_heartbeat_send` falls within the valid byte range `0`–`255`. Setting it to any value **outside** that range — e.g. `-1` — causes the Player to skip sending the heartbeat on every poll tick, indefinitely, with no other change in behavior:

```
{
  "watchdog_heartbeat_send": -1
}
```

This is a deliberate, code-supported mechanism, not an accidental side effect — but it is not a dedicated "heartbeat enable/disable" boolean field; it is a consequence of the byte validity check. **Caution:** disabling the heartbeat removes the PC's only liveness signal entirely. With it disabled, the PC cannot distinguish "the device is healthy but the heartbeat is intentionally off" from "the device has stalled, hung, or crashed" — the Player provides no alternative signal for the PC to fall back on. Only disable this if your deployment's control PC does not rely on heartbeat-based liveness monitoring at all. `watchdog_interval_ms` and `stall_detect_ms` continue to have no effect while the heartbeat is disabled this way — the stall detector still runs internally, it just has nothing to suppress.

Playlist collision validation: at playlist load time, the Player refuses to load a playlist if any item's `on_start_send`, `on_end_send`, or `time_triggers[].send` equals the configured `watchdog_heartbeat_send` **or** `handshake_ack_send` value — a hard, fail-closed refusal, the same pattern used for the reserved receive-code check in §3.8. Without this check, a controller receiving that byte on the wire could not distinguish a heartbeat, or a handshake acknowledgment, from a specific item event that happens to reuse the same value.

3.8 Control command bytes (PC → arm)

Field	Type	Default	Meaning
reboot_receive	int	255	Reboot the device.
stop_receive	int	254	Freeze current frame.
play_right_receive	int	253	Play forward.
play_left_receive	int	252	Play reverse (reverse-paired only).
speed_fast_receive	int	251	Speed 2.0×.
speed_regular_receive	int	250	Speed 1.0×.
speed_slow_receive	int	249	Speed 0.5×.
speed_ultraslow_receive	int	248	Speed 0.25×.

Defaults sit at the top of the byte range (248–255) so they stay clear of ASCII-ish jump codes typically used in `serial_triggers`. Every field above is fully overridable via its matching config.json key — set any of them to any value 0–255 for your deployment.

`hot_reload_receive` does not exist. It was carried in earlier versions of this config but was never actually wired up in the Player. If it appears in an older config.json, it is simply ignored (unknown keys are not errors) — there is no live-reload mechanism; a playlist or config change always requires a reboot.

Per-item **jump** bytes are defined in the playlist's `serial_triggers`, not here.

Validation at load time (playlist-side): the Player refuses to load a playlist if any item's `serial_triggers[].receive` byte equals one of the 8 reserved codes above (using their actual configured values, not just the defaults — an operator may have reassigned them). This is a hard load-time error ("predefined values can not be used in serial_triggers"), not a warning; the whole playlist fails to load, the same fail-closed pattern used throughout the platform (tier gating, reserved send-codes). Practically: pick jump bytes for `serial_triggers` from outside whatever range you've assigned to the 8 fields above.

---

4. Byte Map Summary (defaults)

Dir	Purpose	Byte	Where defined
PC→	Start signature	"HOLOQSTART"	config `serial.start_signature` (bring-up, must complete before anything below works)

Dir	Purpose	Byte	Where defined
→PC	Handshake ack	246	config `serial.handshake_ack_send`
→PC	Heartbeat	247	config `watchdog_heartbeat_send`
→PC	Item on_start	per item	playlist `on_start_send`
→PC	Item on_end	per item	playlist `on_end_send`
→PC	Time trigger	per item	playlist `time_triggers[].send`
PC→	Jump	per item	playlist `serial_triggers[].receive` (must not equal any reserved receive-code below)
PC→	Reboot	255	config `reboot_receive`
PC→	Transport/speed	248–254	config `*_receive`

All single-byte values are 0–255; only the low byte of a command is on the wire. The start signature is the one exception — a short string, matched byte-by-byte.

---

5. Behavioral Rules Tied to Config

- **Link bring-up:** `serial.handshake` defaults to `false` (no gating — behavior unchanged from before this feature existed). If set to `true`, the Player acts on no received byte — `reboot_receive` included — until `serial.start_signature` is matched, at which point it sends `serial.handshake_ack_send` once. See §3.1 and the Serial Protocol Integration Guide §3.1 for the full behavior and timing.
- Heartbeat: emit `watchdog_heartbeat_send` every `watchdog_interval_ms` while healthy. Heartbeat transmission is never gated by the handshake.
- Stall: no progress for `stall_detect_ms` → stop heartbeat, do nothing else (no recovery).
- Reboot: `reboot_receive` → reboot device (the only sanctioned restart), and only once the handshake has completed.
- Command code validation: at playlist load time, the Player checks every item's `serial_triggers[].receive` against the 8 reserved codes in §3.8 (using their actual configured values). Any collision refuses the whole playlist load — see §3.8 for detail.
- Audio: `audio_output` sets `vendor.audio.output.active` at startup (root) and restarts the vendor audio HAL, so only the requested output(s) carry sound; per-item `volume` overrides global. Re-applied on every boot (the property is not persistent).
- Storage: `storage` picks the base dir; removable path found at runtime.

---

6. Example — full config with defaults

See `config.json` shipped alongside this spec (all applicable parameters with their default values).
