HoloQ Player - Playlist Specification (playlist.hlq)

HoloQ Technologies

2026

Contents

HoloQ Player — Playlist Specification (playlist.hlq)

Version: 2.2 Format: JSON (UTF-8). `//` and `/* */` comments and trailing commas are tolerated. Consumed by: HoloQ Player (Android / ExoPlayer).

---

1. Purpose

The playlist defines the ordered set of media items (videos and images), how each item advances, image display duration, reverse-clip pairing, and how serial commands map to jumps and per-item events. Item IDs are assigned by array order (`item_0`, `item_1`, ...). References (`play_start`, `next`, `jump_to`) are by **filename basename**, resolving to the **first** matching item.

> **In practice, use HoloQ Author to create this file.** `playlist.hlq` is a plain JSON file and can technically be hand-written, but HoloQ Author — the companion authoring tool — builds and validates it for you through a visual, graph-based editor, catching malformed or invalid playlists before they ever reach a device. This document specifies the format at a technical level for integrators and anyone who needs to understand it precisely; content producers and show technicians should author playlists in HoloQ Author rather than editing this file by hand. See the HoloQ Author User Manual for details.

---

2. Top-Level Structure

```
{
  "play_start": "<filename>",
  "media_path": "<base dir, relative to the selected storage>",
  "authored_with": "<basic|plus|pro>",
```

```
  "playlist": [ <item>, <item>, ... ]
}
```

Field	Type	Required	Description
play_start	string	yes	Filename of the starting item (resolved by basename).
media_path	string	yes	Base directory for item `file` paths. `"."` = the media root.
authored_with	string	no	Which HoloQ Author tier (`basic`/`plus`/`pro`) last saved this file. Stamped automatically by Author on every save; absent on hand-authored playlists. **Read and enforced by the Player at load time** — see §6.
playlist	array	yes	Ordered items. Order defines `item_0`, `item_1`, …

Path resolution: an item `file` that is an existing absolute path is used as-is; otherwise it resolves under `media_path` within the storage base chosen by `config.storage`.

---

3. Item Object

```
{
  "id": "<optional unique identifier>",
  "type": "video" | "image",
  "file": "<path under media_path>",
  "volume": 100,
  "timeout": 10,
  "next": "<filename or id>",
  "reversed_file": "<path to pre-rendered reverse clip>",
  "on_start_send": 10,
  "on_end_send": 11,
  "serial_triggers": [ { "receive": 97, "jump_to": "<filename or id>" } ],
  "time_triggers":   [ { "elapsed_ms": 3000, "send": 100 } ]
}
```

3.1 Item Fields

Field	Type	Required	Applies to	Default	Description
id	string	no	all	`item_N` (array index)	Unique identifier for this item. If omitted, auto-assigned by array order (`item_0`, `item_1`, …). If provided, **must be unique across the whole playlist** — a duplicate id (whether two explicit ids collide, or an explicit id collides with another item's auto-assigned `item_N`) is refused at load time. See §4.
type	string	yes	all	—	`"video"` or `"image"`.
file	string	yes	all	—	Media path (see resolution above).
volume	int	no	video	100	0–100 for this item. (Reverse-paired items always play muted — see 3.2.)
timeout	int	no	image	5	Seconds to display an image before advancing. Ignored for video.
next	string	no	all	(self)	**id or filename** of the next item (see §4 for resolution order). **Omitted or self → the item loops in place.**

Field	Type	Required	Applies to	Default	Description
reversed_file	string	no	video	none	Path to a pre-rendered reverse clip. Presence makes the item **reverse-paired** (see 3.2).
on_start_send	int	no	all	none	Byte sent to the PC when this item starts.
on_end_send	int	no	all	none	Byte sent to the PC when this item ends/advances.
serial_triggers	array	no	all	[]	Byte → jump mappings active while this item plays (see 3.3).
time_triggers	array	no	video	[]	Send a byte at a given elapsed time within the item (see 3.4). Ignored for reverse-paired.

3.2 Reverse-paired items (`reversed_file`)

When `reversed_file` is present, the item supports forward/reverse transport and speed control: - The item plays **muted** and its **`time_triggers` are ignored** (direction changes make elapsed-time semantics ambiguous). - `reversed_file` must be a frame-exact reversed render of `file`, **same duration and spec**. Direction switches map position `t ↔ D − t` (D = forward duration) for a seamless, frame-accurate flip. - Transport/speed command bytes are defined in `config.json` (stop / play_right / play_left / speed_*), not in the playlist.

3.3 serial_triggers[]

While the current item plays, receiving `receive` triggers an **immediate, seamless** jump to the item whose filename matches `jump_to`.

Field	Type	Required	Description
receive	int	yes	Incoming byte (0–255) that triggers the jump.
jump_to	string	yes	Target item — **id or filename** (resolved per §4).

3.4 time_triggers[] (video only, not reverse-paired)

Send a byte to the PC when playback reaches `elapsed_ms`. Fires once per item play.

Field	Type	Required	Description
elapsed_ms	int	yes	Milliseconds from the start of this item's play.
send	int	yes	Byte (0–255) to send when reached.

---

4. ID Assignment & Reference Resolution

- Every item has an `id`: its explicit `"id"` JSON field if present and non-blank, otherwise auto-assigned by array index (`item_0`, `item_1`, …).
- **Ids must be unique across the whole playlist.** A collision — whether two items provide the same explicit `id`, or an explicit `id` collides with another item's auto-assigned `item_N` — is a **hard load-time refusal**: the whole playlist fails to load, the same fail-closed behavior used for tier-capability and reserved-code checks elsewhere in the platform.
- `play_start`, `next`, `jump_to` are resolved in this order:
  1. **Exact match against an item's `id`.**
  2. If no `id` matches, **fall back to matching by `file` basename**, first match wins.

- This means two items can safely share the same underlying `file` (e.g. the same image shown at two different points in a playlist) as long as they have distinct explicit `id`s — each becomes independently addressable by `next`/`jump_to`. Without distinct `id`s, two items sharing a `file` are **not** independently addressable: any reference to that filename always resolves to whichever of them appears first in the array (see the caution below).
- `next` omitted OR resolving to the item's own id → the item **loops in place** (video replays; image re-holds for `timeout`).
- An unresolved `next` → loop the current item. An unresolved `jump_to` → logged, ignored.

Caution — duplicate `file` values without distinct `id`s: if two items share a `file` and neither has (or only one has) a meaningfully distinct `id`, any reference by that filename still resolves to the first matching item in array order — the second is unreachable by that filename. Always give items sharing a `file` their own distinct, intentional `id`s if both need to be independently addressable.

---

5. Advance & Loop Semantics

- Video with `next`: play to end → fire `on_end_send` → advance to `next`.
- Video without `next`: loop the same video.
- Image with `next`: display `timeout` s → fire `on_end_send` → advance.
- Image without `next`: hold indefinitely.
- Jump (serial): interrupt immediately mid-item and enter the target (fires the target's `on_start_send`). On entering any item: fire `on_start_send`, set `volume` (unless reverse-paired → muted), reset `time_triggers` fired-state, reset speed to 1.0×.
- video→video advances are gapless (2-item window); video→image and image→next are handled on a single reused surface with no black frame.

---

6. Validation Rules

- `type` is one of `video` or `image`.
- Byte fields (`receive`, `send`, `on_start_send`, `on_end_send`) are 0–255.
- `volume` clamped 0–100. `timeout` positive integer for images (default 5).
- `reversed_file`, if present, should exist and match `file`'s duration/spec; a missing reverse file is logged and the reverse switch is ignored (no crash).
- **`id`, if provided, must be unique across the whole playlist.** A duplicate `id` (including an explicit `id` colliding with another item's auto-assigned `item_N`) is a hard load-time refusal — the whole playlist fails to load. See §4.
- Duplicate `file` basenames without distinct `id`s: references resolve to the FIRST — avoid this unless the items are never independently addressed. Give items sharing a `file` distinct `id`s if both need to be reachable by `next`/`jump_to` (§4).
- **`authored_with`, if present, is checked against the device's own tier.** A playlist naming a higher tier than the device is refused entirely at load time — see the Configuration Specification and Products page for the Basic/Plus/Pro capability matrix. If absent, the Player falls back to checking individual gated fields (`on_start_send`/ `on_end_send`, `time_triggers`, `reversed_file`) against the device's tier instead.
- **`serial_triggers[].receive` must not collide with any of the 8 reserved global receive codes** (`reboot_receive`/`stop_receive`/`play_right_receive`/`play_left_receive`/ the four `speed_*_receive` codes, from `config.json`) — a hard load-time refusal if it does. See the Configuration Specification §3.8.
- **`on_start_send`, `on_end_send`, and `time_triggers[].send` must not equal the configured `watchdog_heartbeat_send` or `handshake_ack_send` byte** — a hard load-time refusal if any does.

Without this, a controller receiving that byte on the wire could not distinguish a heartbeat, or the link-bring-up acknowledgment, from a specific item event. See the Configuration Specification §3.1 and §3.7.

---

7. Example (matches the shipped sample)

```
{
  "play_start": "clip1.mp4",
  "media_path": ".",
  "playlist": [
    {
      "type": "video",
      "file": "videos/clip1.mp4",
      "next": "photo1.jpg",
      "on_start_send": 10,
      "on_end_send": 11,
      "time_triggers": [
        { "elapsed_ms": 3000, "send": 100 },
        { "elapsed_ms": 6000, "send": 101 }
      ],
      "serial_triggers": [ { "receive": 97, "jump_to": "clip2.mp4" } ]
    },
    {
      "type": "image",
      "file": "images/photo1.jpg",
      "timeout": 10,
      "next": "clip2.mp4",
      "on_start_send": 20,
      "on_end_send": 21,
      "serial_triggers": [
        { "receive": 98, "jump_to": "clip1.mp4" },
        { "receive": 99, "jump_to": "clip2.mp4" }
      ]
    },
    {
      "type": "video",
      "file": "videos/clip2.mp4",
      "next": "clip3.mp4",
      "on_start_send": 30,
      "on_end_send": 31,
      "serial_triggers": [
        { "receive": 100, "jump_to": "photo1.jpg" },
        { "receive": 101, "jump_to": "clip1.mp4" }
      ]
    },
    {
      // reverse-paired: reversed_file present -> muted, time_triggers ignored.
      // transport bytes (from config, current defaults): stop=254 play_right=253
      // play_left=252; speed_fast=251 speed_regular=250 speed_slow=249 speed_ultraslow=248.
      "type": "video",
      "file": "videos/clip3.mp4",
      "reversed_file": "videos/clip3_reversed.mp4",
```

```
      "next": "clip1.mp4",
      "on_start_send": 40,
      "on_end_send": 41,
      "serial_triggers": [ { "receive": 100, "jump_to": "photo1.jpg" } ]
    }
  ]
}
```

---

8. Implementation Notes

- Images are shown via an overlay ImageView (kept out of the ExoPlayer playlist, which is video-only for reliability), then advanced by a timer.
- Videos should be uniformly encoded so the cedar VPU decodes each cleanly; a 200 ms VPU settle delay is applied on video→video.
- **The playlist is read once at startup; there is no hot-reload.** `hot_reload_receive` was removed entirely in a later release (it was documented in earlier phases but never actually implemented). A change to `playlist.hlq` on storage requires a full device reboot to take effect.

---

9. Change Log

- **id-based resolution** — items may now carry an explicit `"id"` field. `play_start`, `next`, and `jump_to` resolve against explicit ids first, falling back to `file` basename matching only if no id matches. A duplicate id anywhere in the playlist is a hard load-time refusal (§4, §6). This allows two items to safely share the same underlying `file` (e.g. the same image reused at two different points in a branching playlist) as long as each has its own distinct `id`.
- **`authored_with` tier enforcement** — the field is now documented as read and enforced by the Player at load time (§2, §6), not merely stamped by the authoring tool.
- **Start-signature handshake support** — `handshake_ack_send` added to the set of reserved send-codes an item's `on_start_send`/`on_end_send`/`time_triggers[].send` must not collide with (§6). See the Serial Protocol Integration Guide §3.1 and the Configuration Specification §3.1 for the full handshake behavior.