Iziko Constantia — BrightAuthor Presentation Guide

2026-06-02

Preparing a BrightAuthor:Connected presentation — Iziko Constantia

> Step-by-step build guide + hard-won lessons for the 6 BrightSign presentations in the no-reboot, Core-direct architecture. Snapshot **2026-06-02**. Companion to `README.md` (full system handover).

This is the authoring runbook: how to build (or rebuild) one of the six `.bpfx` presentations so it plays in sync, parks silently when off, and is driven entirely by the Q-SYS Core over UDP — no daily reboot.

Table of contents

---

1. How the show works

Each player runs a two-state BrightAuthor:Connected **state machine**: a silent **Black/park** state (the initial state — a per-room ID placard, no video, no markers) and a looping **Show** state (the room's clip). The Q-SYS Core drives the fleet entirely over UDP — `Start` → Show, `Stop` → Black, `Pause` / `Resume` for momentary freezes. The players **never reboot** day-to-day; **roSync** (PTP) frame-locks the videos and re-converges on each `Start`. The **Master** is the roSync Leader and the only player that emits loop markers (`S0` / `L0`) to the Core for audio sync. Every player additionally emits a periodic `SYNC:<id>` heartbeat so the Core can confirm the whole fleet is locked.

---

2. Prerequisites

- **BrightAuthor:Connected** (desktop authoring app).
- **All players on the SAME firmware** — currently **9.1.93.2**. roSync silently refuses to lock across mixed firmware.
- **The 6 videos** (`VIDEOs/UpdatedOptimizedMay2026/Update/b/`), all matched for roSync: **h264, 25 fps, 20402 frames, 816.08 s** (13:36). 5 are 1920×1080 landscape; Office 1 is 1080×1920 portrait. Resolution may differ between players; **frame count and fps must be identical** or roSync won't hold across loop boundaries.
- **The 6 park placards** (`PARK_IMAGES/`): `PARK_M` , `PARK_DR` , `PARK_01` (portrait 1080×1920), `PARK_02` (landscape 1920×1080), `PARK_W1` , `PARK_W2` .
- The **Core** is at `192.168.1.10` and listens on UDP **:7000** (its `sync_receiver.lua`).

---

3. Per-room reference table

Presentation	Player IP	Screen	Video file	Park placard	roSync
`MasterGCUDP`	`192.168.1.11` (LS445)	landscape 1920×1080	`MasterUDP.mp4`	`PARK_M_1920x1080.png`	**Leader**
`DrawingRoomGCUDP`	`192.168.1.30` *	landscape	`DRAWING ROOM with vignette.mp4`	`PARK_DR_1920x1080.png`	Follower
`Office1GCUDP`	`192.168.1.31` *	**portrait 1080×1920**	`Office1_RotatedVideoTrimmed.mp4`	`PARK_O1_1080x1920.png`	Follower
`Office2GCUDP`	`192.168.1.32` *	landscape	`OFFICE 2 with vignette.mp4`	`PARK_O2_1920x1080.png`	Follower
`Window1GCUDP`	`192.168.1.33` *	landscape	`WINDOW 1 with vignette.mp4`	`PARK_W1_1920x1080.png`	Follower
`Window2GCUDP`	`192.168.1.34` *	landscape	`WINDOW 2 with vignette.mp4`	`PARK_W2_1920x1080.png`	Follower

* Follower IP↔room mapping is by physical install — confirm on site. The Master is `.11`; followers occupy `.30–.34`.

Note: Office **1** is the portrait screen (its video is the rotated clip). Office **2** is landscape. The placard orientation must match the screen.

---

4. Build one presentation — step by step

Author all 6 identically except for the per-room values in §3 and the Leader/Follower + marker differences called out below.

4.1 New presentation + screen layout

- Create the presentation. **Screen layout:**
  - Landscape rooms → `Landscape1x1`, canvas 1920×1080.
  - **Office 1 (portrait)** → `Portrait1x1` / `PortraitBottomLeft`, canvas 1080×1920.
- Video mode (HDMI signal) stays `1920x1080x60p` even for the portrait player — BrightSign outputs a landscape signal and rotates the content to the portrait canvas.

4.2 The two states

1. **Black / park = INITIAL state.** Add the room's placard image (§3) as a state and **mark it the initial state**. No video, no audio, no markers — it boots here, silent.
2. **Show state.** Add the room's video. Set **Loop = ON** (`automaticallyLoop`).

4.3 Player Sync (roSync)

- Enable **Enhanced Synchronization** (Player Sync).
- **Master** = Sync **Leader** (`deviceIsSyncMaster = true`); all others = **Follower**.
- **Sync Domain = `6` on ALL SIX.** (A mismatched domain = that player runs in its own group and silently never locks — see Lessons.)

4.4 Networking → UDP (get this exactly right)

On the **Interactive** → **Networking** → **UDP** panel:

Field	Value	Why
UDP Destination Address	**Specific IP address** → `192.168.1.10`	markers/heartbeats go to the Core
UDP Destination Port	**`7000`**	the Core's `sync_receiver` listens here
UDP Receiver Port	**`5000`**	how the Core sends Start/Stop/Pause/Resume to this player

- Do **NOT** select "All players on the local subnet" (that broadcasts).
- Do **NOT** type a netmask (`255.255.255.192`) or `ip:port` (`192.168.1.10:7000`) into the IP field — the port has its own box (see Lessons).

- Followers technically only need the **Receiver Port (5000)** to be controlled; set the destination to the Core anyway for consistency (they emit nothing critical).

4.5 UDP Input List (control receivers — ALL players)

Add these "Specify UDP input" keywords, each wired to an action in §4.6: `Start`, `Stop`, `Pause`, `Resume` (optionally `PING` → reply `PONG`). Leave **"Show in BrightSign App" unchecked** (it only adds a manual phone-remote button; the Core drives everything over UDP regardless).

4.6 Transitions / commands (ALL players)

- **UDP** `Start` → transition to **Show**, restart at **frame 0**, enable **Synchronize** so the fleet enters together.
- **UDP** `Stop` → transition to **Black/park**.
- **UDP** `Pause` → **Pause video** (momentary; no state change).
- **UDP** `Resume` → **Resume video**.

4.7 Markers — MASTER (Leader) ONLY

On the Master's Show state:

- **Video Timecode @ 0 ms** → **Send UDP** `S0` (audio anchor at loop frame 0).
- **Media End** → **Send UDP** `L0` (per-loop backup).

Do **NOT** add `S0`/`L0` to followers — that would send 6× duplicate markers and confuse the Core's audio sync.

4.8 Sync heartbeat — ALL players

On each Show state, add a **Video Timecode event** (one, or several spaced every ~60–120 s of the 816 s loop — more marks = finer monitoring) → **Send UDP** `SYNC:<id>` where `<id>` is the room code: `M` / `DR` / `O1` / `O2` / `W1` / `W2`.

This feeds the Core's per-player sync monitor (`SyncLocked` / `SyncReport` / `SyncSpreadMs`, `sync_receiver.lua` v1.7.0). The keyword is `SYNC:` — deliberately distinct from `S0`/`L0` so it never triggers audio.

4.9 Placard image

Drop the room's park placard (§3) into the Black state. **Orientation must match the screen** — portrait placard on Office 1, landscape on the rest.

---

5. Publish + deploy

- **Publish type =** `Standalone` **/** `Local Storage`**.** NOT `BSN.Cloud`, and NOT "Setup files only" (that produces a "Congratulations, your player is set up!" splash that never plays — see Lessons).
- The global UDP destination baked onto every SD card = `192.168.1.10:7000` (the Core), never the Wall/laptop.
- Deploy via the **Wall** app (or write the SD card directly). Wipe the SD root before redeploying a changed presentation.

---

6. Verify before you ship (bpfx audit)

`.bpfx` files are plain JSON. Before imaging cards, audit all 6 at once. Save as `audit.py` and run `python3 audit.py`:

```
import json, glob, os
base = "/path/to/GC_UDP"           # folder containing the 6 *UDP/*.bpfx
folder = "/path/to/PARK_IMAGES"    # the placard media folder
have = set(os.listdir(folder))
for p in sorted(glob.glob(base + "/*/*.bpfx")):
    d = json.load(open(p, encoding="utf-8", errors="replace")); b = d["bsdm"]
    pr = b["sign"]["properties"]; es = pr.get("enableEnhancedSynchronization", {}) or {}
    ms = b["mediaStates"]["mediaStatesById"]; nm = lambda s: (ms.get(s, {}) or {}).get("name", "?")
    init = [nm(z.get("initialMediaStateId")) for z in b["zones"]["zonesById"].values()]
    refs = set()
    def walk(o):
        if isinstance(o, dict): [walk(v) for v in o.values()]
        elif isinstance(o, list): [walk(v) for v in o]
        elif isinstance(o, str) and o.lower().endswith(".png") and not o.startswith("file:"): refs.add(o)
    walk(b)
    missing = [r for r in refs if r not in have]
    print(os.path.basename(p),
          "dest=%s:%s rx=%s PTP=%s Leader=%s init=%s missing=%s" % (
          pr.get("udpDestinationAddress"), pr.get("udpDestinationPort"),
          pr.get("udpReceiverPort"), es.get("ptpDomain"),
          es.get("deviceIsSyncMaster"), init, missing or "—"))
```

Pass criteria for every player:

- `dest = 192.168.1.10`, **Master** `:7000` (followers `:5000` is fine — they don't emit),
- `rx = 5000`,
- `PTP = 6` on **all six**,
- `Leader = True` on Master only,
- `init` = the room's park placard, and `missing = —` (placard exists in the folder).

Also confirm with `ffprobe` that all 6 videos report the **same** `nb_frames` **and** `r_frame_rate`.

---

7. Lessons learned — the traps we hit

Each of these cost real time on this install. Check them explicitly.

- `255.255.255.192` **in the IP field is a netmask, not an IP.** With "All players on the local subnet" selected, BrightAuthor broadcasts and ignores the IP box — markers never reach the Core. Use **Specific IP** `192.168.1.10`.
- **Don't put the port in the IP field.** `192.168.1.10:7000` typed into the address box won't parse as an IP; the destination port has its own field. IP = `192.168.1.10`, Port = `7000`.
- **PTP / Sync Domain mismatch = silent, intermittent drift.** Window 2 was on domain `0` while the fleet was on `6`; it played but never frame-locked, drifting slowly and looking "randomly off" (re-aligned each reboot, then crept again). **All six must share domain** `6`.
- **roSync needs equal frame count + fps.** Resolution may differ, but a different `nb_frames` or `r_frame_rate` desyncs at loop boundaries. The Master video had to be re-exported to match the fleet's **20402 frames / 25 fps**.
- **The Master plays** `MasterUDP.mp4`, not the Drawing Room clip — confirm the Master's Show state points at the right file.
- **Park placard orientation must match the screen.** Office 1 is **portrait**, so it needs the portrait `PARK_O1` (1080×1920); Office 2 is landscape (`PARK_O2`, 1920×1080). Mismatched orientation = a stretched/rotated placard; mismatched letter = operator confusion.
- **Markers are Leader-only.** `S0` / `L0` on a follower → duplicate markers hammer the Core's audio sync.
- `Receiver Port` **is what lets the Core control a player** (inbound Start/Stop). The destination is outbound (markers). Don't conflate them — a follower with the wrong receiver port simply won't respond to Start/Stop.
- **Publish** `Standalone`**, never "Setup files only".** The latter boots to a "Congratulations, your BrightSign player is set up!" splash and never plays — switch the Destination Type from `BSN.Cloud` to `Standalone` / `Local Storage` and republish.
- **Never replace the BA** `autorun.brs` **with a custom** `Sub Main()`**.** It triggers a black-screen reboot loop (the BrightOS supervisor watchdogs the BA runtime handshake). Author everything inside BrightAuthor; don't hand-edit the runtime.
- **A BrightSign can't power itself off.** "Stop" means transition to the Black state — it's always-on. Display power is a separate concern (CEC/monitor), and display-off does NOT stop playback or markers.
- **Only one Q-SYS component may bind Core** `:7000` (the `sync_receiver`). If the master Lua also binds it, markers split between two listeners ("fires once then nothing").