外部 UDP 控制协议

中控对接指令文档

本文档列出当前版本已实现的全部外部控制指令，含文本写法、等值 JSON、回包规则与现场示例。 所有指令均与界面操作走同一套内部函数，行为完全一致；文中标注的约束均为既有产品逻辑，外控不额外放宽。

24690	39	3	9
控制端口（UDP）	指令条目	支持编码	功能分类

01 通道与通用规则

一条报文一条命令，无连接、无会话。中控只需向目标机固定端口发包即可，不需要握手。

1.1 通道参数

项目	值
传输协议	UDP，无连接
默认端口	`24690`
端口隔离	TCP 24681（Hub 管理）、UDP 24683（设备发现）、UDP 5000（同步组组播）均与本通道无关，互不干扰
配置文件	`data\external_control.json`，默认 `{"enabled":true,"port":24690,"allow":[]}`
来源白名单	`allow` 填 IP **前缀**，空数组表示不限来源
单包上限	8192 字节
报文格式	看首个非空白字符：`{` 为 JSON，其余按纯文本解析
文本关键字	大小写不敏感，`WIN` 与 `win` 等价
编码	UTF-8 / ASCII / GBK 自动识别；回包统一 UTF-8
回包去向	发送方的源 IP 与源端口 —— 中控必须绑定固定端口才收得到回包

1.2 编码自动识别

按固定顺序判定，任一步命中即停止，不会破坏合法的 UTF-8 字节。

顺序	判定条件	处理方式
①	报文带 UTF-8 BOM	先去掉 BOM
②	全部字节小于 0x80	按 ASCII 原样处理
③	字节序列是合法 UTF-8	原样处理
④	以上均不成立	按 GBK（CP936）解码并转为 UTF-8

> 发生 GBK 转换时，日志会记一条 `[外控] 报文按 GBK 解码`（每秒最多一条）。国内老中控、老触摸屏默认就是 GBK 代码页，无需改配置。

1.3 回包规则

命令类别	是否回包	说明
状态类：`status`、`win status`、各类清单命令、`media list`	回包	用于查询与核对序号
其余全部控制类命令（含执行失败）	不回包	失败原因只写日志，不返回错误码

> **对接期请边发边看日志。**控制类命令不回包是按设计如此，不代表没收到。用 `status` 验证链路，用日志 `log\edgeblend_client.log` 的 `[外控]` 行验证执行结果。

1.4 序号口径

位置	起点	示例
文本命令里的 `#N`	**1 起**	`WIN #2` 为第 2 个窗口；`GROUP PLAY #2` 为第 2 个节目
JSON 里的 `index` / `winIndex`	**0 起**	`{"index":1}` 为第 2 个；`{"winIndex":1}` 为第 2 个窗口

换算关系固定：**文本 `#N` = JSON `index` + 1**。编号来源于对应 `list` 回包里的 `index` 字段，现场务必先查后发。

1.5 窗口寻址

写法	含义	备注
`WIN 窗口3`	按窗口名称	**推荐。**现场窗口 ID 不连续（1/236/244/245…），名称最稳
`WIN #2`	按界面顺序第 2 个	与 `status` 回包中的 `windows[].index` 对应，1 起
`WIN 236`	按窗口 ID	写纯数字即按 ID 寻址

写法	含义	备注
省略 `WIN`	基层窗口（窗口 1）	仅 `MEDIA PLAY` 允许省略，其余命令必须写窗口

JSON 侧等价写法：`"win":"窗口3"`、`"winIndex":1`、`"win":236`，三者都不写即基层。

1.6 数值与文件名口径

项目	口径
时间	单位秒，可带小数，如 `12.5`
音量	`0.0 ~ 1.0`，超界自动钳位，`0` 等同静音
音量步长	默认 `0.05`，可显式指定
文件名	只允许 `media\` 下的裸文件名；含 `/ \ :` 或 `..` 一律拒绝
媒体名匹配	含扩展名精确 → 去扩展名 → 唯一前缀；命中多个则拒绝，改用 `#序号`

02 内容源互斥与优先级

同一时刻画面只归一个内容源。发送某一级的开播命令时，客户端会自动收掉冲突的源，而不是静默拒绝——这是现场"发了没反应"问题的根治口径。

3 最高

同步组节目

`group play` `group next` `group prev` 开播时自动停时间轴（含暂停态）并退出播放列表

2 中间

时间轴

`timeline play` `timeline replay` 开播时自动退出播放列表

1 最低

播放列表 / 单窗口媒体

`playlist play|replay|next|prev` `win <w> play|replay` `media play` 开播时自动停时间轴

发出的命令	客户端自动处理
`group play / next / prev`	先停时间轴（暂停态同样停掉）→ 退出播放列表 → 再切节目
`timeline play / replay`	先退出播放列表模式 → 再由时间轴接管画面
`playlist play|replay|next|prev`	先停时间轴 → 再播列表
`media play` `win play` `win replay`	先停时间轴 → 再指定内容

> **为什么必须这样做：**时间轴处于“暂停”状态时，权威标志仍然为真（这是它能“继续/停止”的前提）。旧逻辑下，只是暂停着的时间轴也会把节目与列表命令全部吃掉，现场表现就是“指令发出去毫无反应”。

排错时直接搜这几行日志，一眼确认是谁接管了谁：

```
# 同步组接管（最高优先级）
[外控] 同步组节目接管内容源 ⇒ 先停止时间轴「时间轴 1」@2.94s
# 时间轴接管
[外控] 时间轴接管内容源 ⇒ 退出播放列表模式
# 列表 / 窗口媒体接管
[外控] 媒体/播放列表接管内容源 ⇒ 先停止时间轴「时间轴 1」@9.35s
# 切换过程中不闪黑
[TimeLine] 停止：基层(窗口1)归同步组，不压黑
```

03 通道与状态查询

查询类命令是唯一会回包的一类，也是对接阶段核对序号、确认链路的主要手段。**本章所有命令均回包**，回包目标为发送方的源 IP 与源端口。

功能	文本写法 / 等值 JSON	说明
存活探测	`ping` `{"cmd":"ping"}`	只计数，不验证业务
全量状态	`status` `{"cmd":"status"}`	窗口 / 场景 / 列表 / 时间轴 / 同步组一次全给
单窗状态	`win 窗口3 status` `{"cmd":"win.status","win":"窗口3"}`	当前返回全量窗口表，仅 `cmd` 不同
场景清单	`scene list` `{"cmd":"scene.list"}`	顺序即 `scene #N` 依据
播放列表清单	`playlist list` `{"cmd":"playlist.list"}`	含列表运行态
时间轴清单	`timeline list` `{"cmd":"timeline.list"}`	含时间轴运行态
节目清单	`group list` `{"cmd":"group.list"}`	含节目序号、ID、名称、循环与音量
媒体库清单	`media list [类型]` `{"cmd":"media.list","type":"video"}`	类型：all、video、image、seq、stream、capture、group

```
status
media list
media list video
{"cmd":"media.list","type":"seq"}
```

04 媒体库

不认识文件名也能操作：先 `media list` 拿到序号，再用名称或序号播放。名称匹配支持全名、去扩展名、唯一前缀三种写法。

功能	文本写法 / 等值 JSON	说明
按名称播到窗口	`media play 测试视频.mp4 win 窗口3` `{"cmd":"media.play","name":"测试视频.mp4","win":"窗口3"}`	视频 / 图片走文件路径
不带扩展名	`media play 测试视频 win 窗口3` `{"cmd":"media.play","name":"测试视频","win":"窗口3"}`	去扩展名、唯一前缀均可
按序号播	`media play #3` `{"cmd":"media.play","index":2}`	取 `media list` 的 `index`，0 起
播到基层	`media play 测试视频.mp4` `{"cmd":"media.play","name":"测试视频.mp4"}`	省略窗口即基层（窗口 1）
不循环	`media play 测试视频.mp4 win 窗口2 noloop` `{"cmd":"media.play","name":"测试视频.mp4","winIndex":1,"loop":false}`	默认循环播放
序列帧 / 流 / 采集	`media play 测2 win 窗口4` `{"cmd":"media.play","name":"测2","win":"窗口4"}`	按内容源类型自动打开
媒体组	`media play 组A win 窗口5` `{"cmd":"media.play","name":"组A","win":"窗口5"}`	播组内第一个文件

限制与拒绝规则（会明确报错，不是静默失败）

情形	提示
内容源指定到基层	`基层窗口(窗口1)只支持文件型媒体；内容源请指定 WIN <导播窗口>`
名称前缀命中多个	`名称前缀命中 N 个，请写全名（或用 #序号）`
序号越界	`序号越界（共 N 项，MEDIA LIST 可查）`

media list 回包示例

```
{
  "ok": 1, "cmd": "media.list", "type": "all", "count": 52,
  "items": [
    { "index": 0,  "name": "2.mp4",  "kind": "video", "file": "2.mp4" },
    { "index": 30, "name": "测2",    "kind": "seq",   "src": "_seq/测2", "fps": 25.0, "frames": 30 },
    { "index": 48, "name": "组A",    "kind": "group", "file": "开场.mp4", "items": 3 }
  ]
}
```

`kind` 取值：`video` / `image` / `seq` / `stream` / `capture` / `web` / `group`。

05 窗口控制

窗口级命令覆盖播放、定位、音量、静音与显示隐藏。基层窗口（窗口 1）与导播窗口走同一套语义，仅在"显示/隐藏"上有明确限制。

5.1 播放与定位

功能	文本写法 / 等值 JSON	说明
播文件并循环	`win 窗口3 play 开场.mp4 loop` `{"cmd":"win.play","win":"窗口3","file":"开场.mp4","loop":true}`	只认 media\ 下的文件名
播文件不循环	`win 窗口3 play 开场.mp4 noloop` `{"cmd":"win.play","win":"窗口3","file":"开场.mp4","loop":false}`	播完停在末帧
重播当前素材	`win 窗口3 play` `{"cmd":"win.play","win":"窗口3"}`	不指定文件即沿用当前素材，从头播
暂停	`win 窗口3 pause` `{"cmd":"win.pause","win":"窗口3"}`	画面定格在当前位置
继续	`win 窗口3 resume` `{"cmd":"win.resume","win":"窗口3"}`	从暂停位置继续
重播（回 0）	`win 窗口3 replay` `{"cmd":"win.replay","win":"窗口3"}`	等价于 `play` 不带文件名
停止	`win 窗口3 stop` `{"cmd":"win.stop","win":"窗口3"}`	释放内容并清空该窗口
定位到 12.5 秒	`win 窗口3 seek 12.5` `{"cmd":"win.seek","win":"窗口3","time":12.5}`	暂停中也可定位

5.2 音量与静音

功能	文本写法 / 等值 JSON	说明
设定音量	`win 窗口3 vol 0.8` `{"cmd":"win.volume","win":"窗口3","value":0.8}`	别名 `volume`
音量加	`win 窗口3 volup 0.1` `{"cmd":"win.volup","win":"窗口3","step":0.1}`	步长可省，默认 0.05
音量减	`win 窗口3 voldown 0.1` `{"cmd":"win.voldown","win":"窗口3","step":0.1}`	步长可省，默认 0.05
静音开	`win 窗口3 mute on` `{"cmd":"win.mute","win":"窗口3","on":true}`	`mute` 单独写也等于开
静音关	`win 窗口3 mute off` `{"cmd":"win.mute","win":"窗口3","on":false}`	恢复原音量
静音取反	`win 窗口3 mute toggle` `{"cmd":"win.mute","win":"窗口3","toggle":true}`	中控面板单键切换常用

> 音量与静音改动会落盘到 `data\director_windows.json`（节流 1.5 秒），重启后保持。

5.3 窗口开启 / 关闭

功能	文本写法 / 等值 JSON	说明
显示	`win 窗口3 show` `{"cmd":"win.visible","win":"窗口3","on":true}`	立即恢复画面
隐藏	`win 窗口3 hide` `{"cmd":"win.visible","win":"窗口3","on":false}`	不停播、不卸载内容
显式设值	`win 窗口3 visible off` `{"cmd":"win.visible","win":"窗口3","on":false}`	`on/off` 可省略
取反	`win 窗口3 visible toggle` `{"cmd":"win.visible","win":"窗口3","toggle":true}`	走场切换常用

> **基层窗口不支持显示 / 隐藏。**它是融合底图，渲染端跳过其 visible 判定。该命令会被明确拒绝并提示：`基层窗口(窗口1)是融合底图，不支持显示/隐藏；要黑场请用 TIMELINE STOP 或 WIN 1 STOP`。

隐藏语义只改 `visible` 字段并落盘；隐藏后客户端在下一帧释放该窗口播放器与纹理，再次显示时重新加载 —— 属既有行为，不额外增加延迟。

06 场景

应用场景等于整份布局快照替换（含窗口位置、内容、可见性），并落盘、回传 Hub。这与在界面里点场景是同一个函数。

功能	文本写法 / 等值 JSON	说明
按名称切场景	`scene 开场` `{"cmd":"scene.apply","name":"开场"}`	等价于界面点场景
显式写法	`scene apply 开场` `{"cmd":"scene.apply","name":"开场"}`	与上一条完全等价
按序号切场景	`scene #2` `{"cmd":"scene.apply","index":1}`	取 `scene list` 顺序，0 起
场景清单	`scene list` `{"cmd":"scene.list"}`	回包，用于核对序号

07 播放列表

列表的"图片停留时长 / 视频播完自动下一项"由客户端既有调度推进，外控不改变该节奏；外控只负责选列表、人工上下条与整体暂停。

功能	文本写法 / 等值 JSON	说明
按名称播放	`playlist play 列表A` `{"cmd":"playlist.play","name":"列表A"}`	从第 1 项开始

功能	文本写法 / 等值 JSON	说明
按序号播放	`playlist play #1` `{"cmd":"playlist.play","index":0}`	取 `playlist list` 顺序
设待机列表	`playlist standby 列表A` `{"cmd":"playlist.standby","name":"列表A"}`	落盘，不改变当前在播
下一条	`playlist next` `{"cmd":"playlist.step","delta":1}`	列表内循环，缺文件的项自动跳过
上一条	`playlist prev` `{"cmd":"playlist.step","delta":-1}`	反向逐个尝试，规则同上
暂停	`playlist pause` `{"cmd":"playlist.pause"}`	冻结自动推进 + 视频项暂停播放器
继续	`playlist resume` `{"cmd":"playlist.resume"}`	解除冻结，暂停时长不计入停留
重播列表	`playlist replay` `{"cmd":"playlist.replay"}`	当前列表从第 1 项重来；无在播则用待机列表
重播指定列表	`playlist replay 列表A` `{"cmd":"playlist.replay","name":"列表A"}`	同时切换并重播
退出列表模式	`playlist stop` `{"cmd":"playlist.stop"}`	交出基层控制权
列表清单	`playlist list` `{"cmd":"playlist.list"}`	回包，含运行态

pause / resume 的真实语义

- 冻结的是**列表推进**，不只是视频播放器
- 图片项：停留计时一起冻结，恢复后接着计时
- 视频项：同时暂停播放器
- 时间轴接管期间发暂停，只冻结列表，画面仍归时间轴

next / prev 的容错

- 沿方向逐个尝试，最多走完整圈
- 文件不存在或打不开的项直接跳过
- 整圈都不可播才记一条失败日志
- 不会出现"按了没反应"的静默失败

08 时间轴

时间轴接管期间，画面由时间轴调度，窗口级播放/暂停会被忽略——这是设计口径，不是指令失效。

功能	文本写法 / 等值 JSON	说明
从 0 播放	`timeline play` `{"cmd":"timeline.play"}`	使用当前活跃时间轴
指定名称播放	`timeline play T1` `{"cmd":"timeline.play","name":"T1"}`	
指定序号播放	`timeline play #2` `{"cmd":"timeline.play","index":1}`	取 `timeline list` 顺序
指定起点播放	`timeline play T1 at 12.5` `{"cmd":"timeline.play","name":"T1","time":12.5}`	`time` 为起播位置
暂停	`timeline pause` `{"cmd":"timeline.pause"}`	定格，权威标志仍为真
继续	`timeline resume` `{"cmd":"timeline.resume"}`	从暂停位置接着播
重播	`timeline replay` `{"cmd":"timeline.replay"}`	当前时间轴从 0 重来
重播指定	`timeline replay #2` `{"cmd":"timeline.replay","index":1}`	切换并重播
停止	`timeline stop` `{"cmd":"timeline.stop"}`	未在播时忽略，不会误压黑
定位	`timeline seek 30` `{"cmd":"timeline.seek","time":30}`	暂停中也可拖
时间轴清单	`timeline list` `{"cmd":"timeline.list"}`	回包，含运行态

> **停止的后果：** `timeline stop` 会把基层压黑，这是时间轴权威期结束的正确表现。要恢复常规播放，请随后发 `win 1 play`、`media play` 或 `scene`。若基层已被同步组或播放列表接管，停止时间轴不会压黑。

09 同步组节目

同步组是所有内容源里优先级最高的一类。节目切换必须由主机发起，客户端会转交 Hub 统一下发，保证主从同一执行时刻。

功能	文本写法 / 等值 JSON	说明
按序号播节目	`group play #2` `{"cmd":"group.program.play","index":1}`	取 `group list` 顺序
按名称播节目	`group play 开幕式` `{"cmd":"group.program.play","name":"开幕式"}`	
按节目 ID 播	`group play 3` `{"cmd":"group.program.play","id":3}`	纯数字按节目 ID 寻址
下一个节目	`group next` `{"cmd":"group.program.step","delta":1}`	按节目顺序循环

功能	文本写法 / 等值 JSON	说明
上一个节目	`group prev` `{"cmd":"group.program.step","delta":-1}`	反向，规则同上
暂停	`group pause` `{"cmd":"group.pause"}`	整组同时暂停
继续	`group resume` `{"cmd":"group.resume"}`	从暂停处继续
重播	`group replay` `{"cmd":"group.replay"}`	当前节目从 0 重播，主从各按本机文件
节目清单	`group list` `{"cmd":"group.list"}`	回包，含节目序号、ID、名称、循环与音量

9.1 执行路径

情形	走法	结果
主机在线、Hub 在线	客户端 → Hub `program_play_req` / `program_ctrl_req` → 逐台 TCP 下发 + 组播统一执行时刻（服务端时钟 + 250 毫秒）	主从同刻动作，不会一前一后
主机在线、Hub 离线	本地直接执行，从机靠主机锚点与同步命令跟随	仍可切换，精度低于统一时刻
从机	拒绝执行，只记一条日志	日志：本机是从机，同步组控制请发给主机

> `group play` 转交 Hub 后，JSON 里的 `loop` / `volume` 不生效，以节目自身配置为准——与界面点按完全一致。若 Hub 未在 1.5 秒内真正执行（例如该节目没给本机配媒体文件），客户端会本地兜底补切，日志记 `[外控] Hub 未在 1.5s 内执行节目切换 ⇒ 本地兜底执行`。

10 状态回包字段

`status` 与 `win status` 返回同一结构，是现场核对运行态的唯一权威来源。

字段	类型	说明
`ok`	整数	1 表示正常
`cmd`	字符串	`status` 或 `win.status`
`app` / `version`	字符串	客户端标识，诊断用
`windows[]`	数组	窗口表，字段见下
`windows[].id`	整数	窗口 ID，现场不连续

字段	类型	说明
`windows[].index`	整数	界面顺序，0 起；文本 `#N` = index + 1
`windows[].name`	字符串	窗口名称
`windows[].file`	字符串	当前内容：文件型为文件名，内容源为显示名，基层为主播放器内容
`windows[].kind`	字符串	`file / seq / stream / capture / web`
`windows[].visible`	布尔	是否显示；基层恒为 true
`windows[].playing`	布尔	是否在播
`windows[].time` / `duration`	小数	当前进度 / 时长，单位秒
`windows[].volume`	小数	0 ~ 1
`windows[].muted`	布尔	是否静音
`scenes[]`	数组	场景名称列表，顺序即 `scene #N`
`playlists[]`	数组	播放列表名称，顺序即 `playlist play #N`
`playlist`	对象	`{playing,index,standby,mode,paused}`；`paused=true` 表示列表已暂停
`timelines[]`	数组	时间轴名称，顺序即 `timeline play #N`
`timeline`	对象	`{name,playing,time,authority}`；`authority=true` 表示时间轴接管画面
`sync`	对象	`{group,role,program}`；`role` 为 `master` / `slave`
`groupPrograms[]`	数组	`{index,id,name,loop,volume}`，顺序即 `group play #N`

11 典型场景示例

11.1 开机起播（基层 + 两个导播窗口）

```
win 1 play 开场.mp4 loop
win 窗口2 play 副屏.mp4 loop
win 窗口3 play 循环背景.mp4 loop
win 1 vol 0.6
win 窗口2 vol 0.3
win 窗口3 mute on
```

11.2 媒体库调用（不认识文件名也能操作）

```
media list video
media play 测试视频 win 窗口3
media play #3 win 窗口4 noloop
media play 测2 win 窗口5
```

```
{"cmd":"media.play","name":"测试视频","win":"窗口3","loop":true}
{"cmd":"media.play","index":2,"winIndex":3,"loop":false}
```

11.3 走场时临时收起某个画面

```
win 窗口5 hide
win 窗口5 show
win 窗口6 visible toggle
```

11.4 切场景后由时间轴接管

```
scene 开场
timeline play 主时间轴 at 0
timeline pause
timeline resume
timeline seek 30
timeline replay
timeline stop
```

11.5 播放列表排播

```
playlist standby 暖场
playlist play 暖场
playlist next
playlist prev
playlist pause
playlist resume
playlist replay
playlist stop
```

11.6 同步组节目（主从同刻切换）

```
group list
group play #1
group play 开幕式
group next
group pause
group resume
group replay
```

11.7 GBK 中控（Python 示例）

```python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind(("0.0.0.0", 8800))          # 绑定固定端口才能收回包
s.settimeout(2.0)

# 老中控发 GBK 也能被正确识别
s.sendto("win 窗口3 play 开场.mp4 loop".encode("gbk"), ("192.168.5.102", 24690))

s.sendto("status".encode("ascii"), ("192.168.5.102", 24690))
data, addr = s.recvfrom(65535)
print(data.decode("utf-8"))
```

11.8 收包型中控（JSON 一次下发多条）

```
{"cmd":"win.play","win":"窗口2","file":"副屏.mp4","loop":true}
{"cmd":"win.volume","win":"窗口2","value":0.45}
{"cmd":"scene.apply","index":2}
{"cmd":"timeline.play","name":"主时间轴","time":0}
```

12 日志与排错

客户端日志：`log\edgeblend_client.log`；Hub 日志：`log\sync_group_debug.log`。

12.1 关键日志行

日志行	含义
`[外控] UDP 控制通道已监听 0.0.0.0:24690（来源不限）`	通道已起，端口与白名单见该行
`[外控] win.play ← 192.168.5.102:8800`	报文已到达并成功解析成命令
`[外控] 报文无效(...): 未知命令: XXX`	关键字写错，裸命令一般不被识别
`[外控] 报文按 GBK 解码(...)`	编码自动转换生效
`[外控] 媒体播放失败: ...`	媒体名或序号不对、内容源进了基层等
`[外控] 已请求 Hub 下发节目 (...)`	`group play` 已转交 Hub 统一下发
`[外控] 已请求 Hub 执行同步组 pause/resume/replay`	整组控制已转交 Hub
`[外控] Hub 未在 1.5s 内执行节目切换 ⇒ 本地兜底执行`	Hub 侧无法执行，客户端补切
`[SyncGroupMgr] 播放节目 N 'xx' devices=2`	Hub 已按统一下发路径执行（外控转交成功）

12.2 常见问题对照

现象	原因	正确做法
发 `ping` 没反应	`ping` 按设计不回包	用 `status` 验证链路
发 `voldown` 没反应	裸命令不被识别	`win 窗口3 voldown 0.05`
发 `stop` 没反应	裸命令不被识别	`win 窗口3 stop` / `playlist stop` / `timeline stop`
发 `prev` 没反应	裸命令不被识别	`playlist prev` / `group prev`
`scene #2` 没生效	序号越界，场景不足 2 个	先 `scene list` 核对
`media play X` 报前缀多个	名称存在歧义	写全名，或改用 `media play #N`
中文命令乱码或无效	编码既非 UTF-8 也非 GBK	统一 UTF-8（推荐）
时间轴在播时窗口命令无效	`timeline.authority=true`	先 `timeline stop`，或直接发更高优先级命令
从机发 `group xxx` 被拒	节目与整组控制必须由主机发起	把命令发给主机 IP
收不到回包	中控没有绑定固定端口	先 `bind()` 再 `sendto()`

13 与界面操作的一致性

外控命令不比界面操作多任何特权：时间轴权威、桌面融合禁用、同步组主从身份等既有约束一律照常生效，因此不会触碰同步算法。

外控命令	内部执行	界面侧
`win.play/pause/resume/stop/seek`（基层）	主播放器对应函数	同一组函数
`win.*`（导播窗口）	导播窗口对应处理	同一组函数
`win.volume/mute/visible`	写窗口字段并落盘	同一组字段，差 1 帧生效
`media.play`	文件型走窗口播放；内容源写窗口类型与路径	与界面推送、拖拽同一口径
`scene.apply`	场景应用	同一函数
`playlist.*`	列表起播 / 待机 / 上下条 / 暂停 / 继续	同一组函数
`timeline.*`	时间轴传输控制	同一函数

外控命令	内部执行	界面侧
`group.play/step`	Hub 节目播放	同一函数
`group.pause/resume/replay`	Hub 整组控制	同一函数

> 唯一例外是内容源开播命令会按第二章的优先级自动收掉冲突源。不这么做的话，画面会被上一级立刻盖回，外控看起来完全无效。

版本记录

版本	日期	变更
v3.2	2026-10-18	内容源互斥与优先级：同步组 3 > 时间轴 2 > 列表 / 窗口媒体 1；切内容时不再闪黑
v3.1	2026-10-18	修播放列表 next / prev / pause / resume / replay：缺文件项自动跳过，暂停改为列表语义，开播前先停时间轴
v3	2026-10-18	新增窗口开关、媒体库查询与调用、场景/列表/时间轴序号寻址、列表与时间轴的暂停/继续/重播、同步组整组控制、GBK 自动识别
v2	2026-10-18	节目切换改走 Hub 统一下发，主从同刻切换
v1	2026-10-18	通道、窗口、场景、列表、时间轴、同步组基础命令