elevator-ad-platform

详细模块设计与关键函数

1. 端侧：智能播放终端 (Go + Vue + python)

目标：极致的稳定性（7x24不崩溃）、流畅的视频播放、断网也能工作。 推荐库：FFmpeg (解码), Qt/SDL (渲染), SQLite (本地库), CURL (下载)。

A. 播放引擎模块 (Player Engine)

底层的视频加载与渲染。

`void InitVideoHardware()`: 初始化显卡加速（如硬解），设置屏幕分辨率和旋转角度（电梯屏常为竖屏）。

`bool LoadMedia(string filepath)`: 预加载视频文件到缓冲区，检查文件完整性（MD5校验）。
`void RenderLoop()`: 主渲染循环。
功能: 每一帧的绘制，处理淡入淡出转场效果。同时监听从云发来的指令

`void SetOverlay(string text_content)`: 在视频上方渲染滚动字幕（跑马灯）或紧急通知图层。

B. 调度与队列模块 (Schedule Manager)

广告播放队列的维护

```
素材库：
"assets": [
    {
    "id": "AD_NIKE_01",
    "type": "video",
    "filename": "nike_2026_q1.mp4", // 拼接 base_url 使用
    "md5": "a1b2c3d4e5f6...",       // 下载后必须校验此值
    "duration": 15,                 // 预计时长(秒)
    "size_bytes": 10485760          // 文件大小(10MB)
    },
    {
    "id": "AD_COKE_CNY",
    "type": "video",
    "filename": "coke_cny_final.mp4",
    "md5": "f9e8d7c6b5...",
    "duration": 15,
    "size_bytes": 5242880
    },
    {
    "id": "AD_公益_01",
    "type": "image",
    "filename": "public_welfare.jpg",
    "md5": "123456...",
```

```
    "duration": 10,                 // 图片展示10秒
    "size_bytes": 512000
    }
]
```

.

```
void SyncSchedule(string json_schedule):
功能: 解析从云端下发的JSON播放清单，存入本地SQLite数据库。
逻辑: 将时间段（如 8:00-9:00）映射到具体的广告列表。
{
    "policy_id": "POL_SH_20260112_V1", // 策略包ID，用于版本比对
    "effective_date": "2026-01-12",    // 生效日期
    "download_base_url": "https://oss.cdn.com/ads/", // 素材下载公共前缀

    "global_config": {
        "default_volume": 60,       // 默认音量 (0-100)
        "download_retry_count": 3,  // 素材下载失败重试次数
        "report_interval_sec": 60   // 日志上报频率
    },

    // ========================================
    // 2. 紧急插播/霸屏策略 (Interrupts)
    // 优先级最高 (Priority 0)，无视任何时间段
    // 场景：消防报警、物业紧急通知、VIP包断
    // ========================================
    "interrupts": [
        {
        "trigger_type": "command", // 触发方式: command(指令触发), signal(硬件信号)
        "ad_id": "AD_EMERGENCY_FIRE",
        "priority": 999,           // 绝对高优
        "play_mode": "loop_until_stop" // 循环播放直到收到停止指令
        }
    ],

    "time_slots": [
        {
        // [早高峰策略] 优先级高
        "slot_id": 1,
        "time_range": "08:00:00-10:00:00", // 时间段
        "volume": 80,              // 人多嘈杂，音量大
        "priority": 10,                    // 优先级 (数字越大越高)
        "loop_mode": "sequence",           // 播放模式: sequence(顺序), random(随
机)
        "playlist": ["AD_NIKE_01", "AD_COKE_CNY"] // 引用下方的素材ID
        },
        {
        // [晚高峰策略]
        "slot_id": 2,
        "time_range": "17:00:00-19:00:00",
        "volume": 50,
        "priority": 10,
        "loop_mode": "sequence",
        "playlist": ["AD_NIKE_01", "AD_APP_PROMO"]
        },
        {
        // [默认兜底策略] 当上述时间段都不匹配时，播这个
```

```
        "slot_id": 99,
        "time_range": "00:00:00-23:59:59",
        "volume": 0,
        "priority": 1,
        "loop_mode": "random",
        "playlist": ["AD_公益_01", "AD_LOGO_01"]
        }
    ]
}
Asset GetNextAsset():
功能：根据当前系统时间、优先级算法，计算下一个该播放的文件。
逻辑：处理插播（紧急）、轮播（普通）、定投（特定时间）。

void CleanupStorage()：磁盘空间管理，LRU算法删除过期的广告文件。
```

C. 监控与数据上报 (Monitor & Logger)

上报播放历史

```
增加用户交互的历史
void RecordPlayLog(string ad_id, timestamp start, timestamp end):
功能：记录每一次播放的开始和结束时间，写入本地加密日志文件（防篡改）。
{
    "type": "play_logs",
    "logs": [
        {"ad_id": "ad_101", "start": 1709810000, "duration": 15},
        {"ad_id": "ad_102", "start": 1709810015, "duration": 30}
    ]
}
string CaptureScreenSnapshot():
功能：调用底层API截取当前屏幕画面（作为远程监控的凭证），压缩为JPEG。
{
    "type": "snapshot_response",   // 消息类型：截图响应
    "did": "ELEV_001",             // 设备ID
    "ts": 1709812399,              // 截图产生的时间戳
    "req_id": "cmd_8821_xc",       // 关联ID：回应的是云端哪一条指令
    "payload": {
        "format": "jpeg",            // 图片格式：jpeg 体积最小
        "quality": 60,               // 压缩质量：60%（监控够用了）
        "resolution": "640x360",     // 分辨率：缩略图尺寸
        "data": "/9j/4AAQSkZJRgABA..." // 图片的 Base64 编码字符串（非常长）
    }
}
HealthStatus GetSystemHealth():
功能：收集CPU温度、内存占用、磁盘余量、网络信号强度 (RSSI)。
    {
    "type": "heartbeat",    //消息类型
    "did": "ELEV_001",      //Device ID
    "ts": 1709812345,       //时间戳 Timestamp
    "status": {
        "cpu": 15,         // CPU负载
        "mem": 40,         // 内存占用
        "playing": "ad_88",// 当前正在播放的广告ID
        "ver": "1.2.0"     // 软件版本
        }
    }
```

D. 自动维护

守护进程逻辑 (Guardian.cpp)

```
#include <sys/wait.h>
#include <unistd.h>

void GuardianLoop() {
    while (true) {
        pid_t pid = fork();

        if (pid == 0) {
            // 子进程：执行主程序
            execl("/usr/bin/ad_player", "ad_player", NULL);
        } else if (pid > 0) {
            // 父进程：监控子进程
            int status;

            // 设定一个非阻塞的检查，或者配合心跳机制
            // 这里演示最简单的：等待子进程退出
            waitpid(pid, &status, 0);

            // 代码运行到这里，说明主程序挂了！

            // 1. 记录崩溃时间到本地文件
            WriteCrashLog("crash_detected", GetCurrentTime());

            // 2. 简单的指数退避策略（防止无限快速重启）
            sleep(3);

            // 3. 循环继续，自动再次 fork 启动
            std::cout << "Main app crashed! Restarting..." << std::endl;
        }
    }
}
```

错误上报

```
void OnSystemInit()
重启后检查日志并上报到云
```

2. 通信层：高性能接入网关 (C++)

目标：承载成千上万台电梯终端的并发连接，作为端与云的“路由器”。 协议：推荐自定义协议 over TCP 或 WebSocket（为了双向通信）。

A. 连接管理 (Connection Manager)

```
func ListenAndServe(port string): 启动TCP/WebSocket监听。

func HandleHandshake(conn Connection) error:
功能: 处理端侧的鉴权（DeviceID + Token），验证合法性，建立Session。

func KeepAliveManager():
功能: 维护心跳包（Heartbeat）。如果在N秒内未收到端侧心跳，判定设备离线，并通知Python后端更新状态。/设备重启
```

B. 消息路由 (Message Router)

```
func DispatchMessage(msg []byte):
功能: 解析端侧发来的包。如果是“日志”，写入Kafka/数据库；如果是“截图”，上传OSS并回调Python；如果是“状态查询”，直接回复。

func PushCommand(deviceID string, cmd Command):
功能: 供Python后端调用（通过gRPC/HTTP接口）。查找该deviceID对应的内存中的Connection，将指令（如“更新播放列表”）通过长连接下发给C++端。
```

3. 云端：业务控制中心 (Go + vue + python)

目标：复杂的业务逻辑、Web界面、数据统计。 框架：FastAPI 或 Django。

A. 设备与内容管理 (Device & CMS)

```
def register_device(device_info): 设备入库，生成唯一的证书/Token。

def upload_material(file, metadata):
功能: 广告主上传视频。
逻辑: 触发转码任务（统一转为端侧支持的格式，如H.264 MP4 1080p），生成MD5。
同时维护一个数据库
{
    "ad_master":"red_bull"          //广告主名字
    "id": "AD_NIKE_01",
    "type": "video",
    "filename": "nike_2026_q1.mp4", // 拼接 base_url 使用
    "md5": "a1b2c3d4e5f6...",       // 下载后必须校验此值
    "duration": 15,                 // 预计时长(秒)
    "size_bytes": 10485760          // 文件大小(10MB)
    },

def create_campaign_strategy(ads_list, devices_list, time_rules):
功能: 制定投放策略。例如：“广告A在北京市所有写字楼电梯，早高峰8:00-10:00循环播放”。
输出: 生成一份JSON格式的 schedule_config。
{
    "type": "schedule_update",
    "version": "20260107_v1",
    "download_base_url": "https://oss.aliyun.com/ads/",
    "playlist": [
        {
        "id": "ad_101",         //广告id
        "file": "coke_cny.mp4",//广告文件名
        "md5": "a1b2...",       //完整性校验码
        "priority": 10,       // 优先级
        "slots": ["08:00-10:00", "17:00-19:00"] // 播放时段
```

```
        },
        {
        "id": "ad_102",
        "file": "nike_shoe.mp4",
        "md5": "c3d4...",
        "priority": 5,
        "slots": ["*"]        // 全天轮播
        }
    ]
}
```

B. 远程控制与监控 (Remote Ctrl)

```
def send_remote_command(device_id, command_type):
功能: 调用Go服务的接口。指令包括：重启设备、更新音量、截屏、强制插播。
    {
        // ------------------------------------------------
        // 1. 信封头 (Header)
        // ------------------------------------------------
        "cmd_id": "CMD_8829_XC92",      // [指令ID] UUID，用于追踪执行结果（异步回调）
        "target_did": "ELEV_001",       // [目标设备] 指定哪台电梯执行
        "send_ts": 1709825000,          // [下发时间] 用于判断指令是否过期
        "expire_sec": 60,               // [过期时间] 如果60秒内发不下去（设备离线），则
丢弃指令

        // ------------------------------------------------
        // 2. 动作定义 (Action)
        // ------------------------------------------------
        "action": "set_volume",         // [动作类型] 枚举值: reboot, set_volume,
capture, insert_play

        // ------------------------------------------------
        // 3. 参数载荷 (Payload)
        // 根据 action 不同，这里的结构也不同 (见下文详情)
        // ------------------------------------------------
        "params": {
            // ... 具体参数
        }
    }

def get_device_snapshot(device_id): 请求实时截屏并在前端展示。
```

纯文本紧急通知:

```
    "params": {
    "type": "text_overlay",     // 插播类型: 文字图层
    "content": "电梯即将进行维保，请勿乘坐", // 文字内容
    "duration": 60,             // 显示时长(秒)
    "font_size": 48,
    "color": "#FF0000",         // 红色警示
    "position": "center"        // 居中显示
    }
```

强制切播视频

```
"params": {
"type": "video_override",  // 插播类型: 视频霸屏
"ad_id": "AD_FIRE_SAFETY", // 必须是端侧已经下载好的素材ID
"loop_times": 3,           // 强制循环播放3次
"resume_policy": "restart" // 插播结束后：restart(重新开始原节目), continue(继续原进度)
}
```

C. 历史数据与报表 (Analytics)

```
def process_play_logs(log_batch):
功能: 消费端侧上报的日志。
逻辑: 对比计划表与实际播放日志，计算“完播率”。
{

    "log_id": "9f8c12d4-55e1-4a2b-9c8d-123456789abc",                    // [日志唯一标识]，用于日志去重
    "device_id": "ELEV_SH_001",                                          // [设备ID] 对应电梯终端的硬件编号
    "ad_id": "AD_NIKE_2026_CNY_V1",                                      // [广告ID]

    "playback_info": {
        "start_time": "2026-01-11T08:30:00.000+08:00",                   // [开始时间]
        "end_time": "2026-01-11T08:30:15.000+08:00",                     // [结束时间]
        "duration_ms": 15000,                                            // [实际播放时长]
        "status_code": 0,       // [机器状态码] 0:正常结束, 1:意外中断, 2:被系统切播, 3:解码失败
        "status_msg": "completed"        // [状态描述] 人类可读的状态说明
    },

    "security_check": {
        "expected_md5": "a1b2c3d4e5f67890a1b2c3d4e5f67890",// [预期MD5] 云端下发排期表时指定的正确文件哈希值
        "actual_md5": "fffff3d4e5f67890a1b2c3d4e5f6xxxx" // [实际MD5] 端侧在播放时实时计算的文件哈希值
    },

    // ----------------------------------------------------
    // 4. 审计结果 (Audit Result) - 云端计算字段
    "audit_result": {
        "is_valid": false,      // [是否有效] true: 计入广告费; false: 不计费（坏账）
        "billing_status": "ignored", // [计费状态] ignored(忽略/不计费), billed(已计费), pending(待人工复核)
        "note": "端侧上报MD5与排期表预设MD5不一致，判定为素材损坏或篡改"// [备注说明] 详细解释原因，本例中是MD5校验失败
    },

    // ----------------------------------------------------
    // 5. 元数据 (Meta)
    // 辅助运维排查问题的信息
    // ----------------------------------------------------
    "meta": {
```

```
        // [客户端IP] 记录上报时的网络出口IP，用于地理位置校验
        "client_ip": "192.168.1.105",

        // [固件版本] 记录播放时端侧软件的版本号，用于排查特定版本的Bug
        "firmware_version": "v2.1.0",

        // [入库时间戳] Unix时间戳，即云端服务器收到这条日志的时间
        // 用于监控网络延迟（created_at - end_time = 延迟秒数）
        "created_at": 1709815005
    }
}
```

def generate_billing_report(client_id, month): 基于完播数据，生成广告主的结案报告（共播放多少次，覆盖多少人次）。