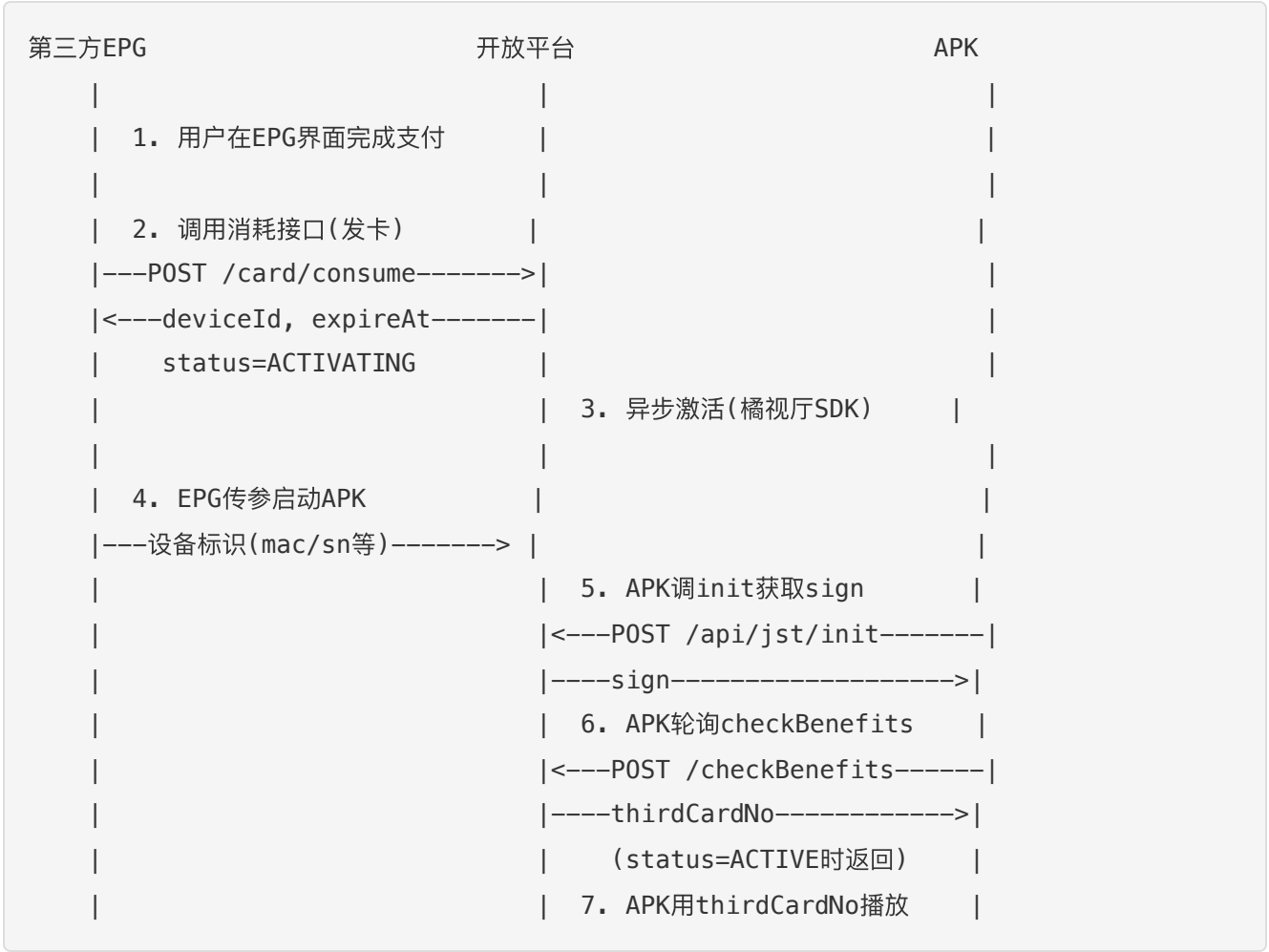
天卡开放平台 第三方对接文档

版本: V2.0
更新日期: 2026-07-23

1. 概述

本文档面向第三方合作伙伴后端开发人员，描述天卡开放平台的接口规范。

1.1 业务流程



1.2 服务器地址

环境	地址
生产环境	http://58.220.83.58:8084

---

2. 鉴权方式

2.1 凭证

平台分配的 `appKey` 和 `appSecret`，由管理后台创建合作伙伴时自动生成。

2.2 签名算法

```
sign = MD5(appKey + timestamp + nonce + appSecret).toUpperCase()
```

- 拼接顺序固定：appKey + timestamp + nonce + appSecret，无分隔符
- MD5 结果转大写

2.3 请求头

每个请求必须携带以下 HTTP 请求头：

请求头	类型	说明
appKey	String	平台分配的应用 Key
timestamp	String	毫秒级时间戳，有效期 5 分钟
nonce	String	随机字符串，5 分钟内不可重复
sign	String	签名（大写 MD5）

2.4 鉴权失败响应

HTTP 401：

```
{"code": 401, "msg": "签名校验失败"}
```

2.5 统一响应格式

```
{"code": 0, "msg": "success", "data": {}}
```

- `code = 0` 表示成功，非 0 表示失败
- 业务数据在 `data` 字段内

---

3. 天卡代销接口

3.1 消耗天卡

用户在 EPG 完成支付后调用，从合作伙伴天卡池分配 1 张天卡绑定到设备，并异步激活。

```
POST /openapi/partner/card/consume
Content-Type: application/json
```

请求体：

参数	类型	必填	说明
mac	String	否*	设备 MAC 地址
wifiMac	String	否*	WiFi MAC 地址
deviceUid	String	是*	设备唯一标识
deviceSn	String	否*	设备序列号
channelOrgId	Long	否	渠道组织 ID

* mac / wifiMac / deviceUid / deviceSn deviceUid必须传，mac / wifimac /deviceSn 至少传一个，需与 APK 启动时传入的设备标识一致。

请求示例：

```
{
  "mac": "AA:BB:CC:DD:EE:01",
  "deviceSn": "SN-001",
  "deviceUid": "UID-001"
}
```

响应：

字段	类型	说明
deviceId	Long	设备 ID
expireAt	String	天卡过期时间（yyyy-MM-dd HH:mm:ss），初始为 24 小时，异步激活成功后更新为真实过期时间
status	String	天卡状态：ACTIVATING（激活中）

响应示例：

```
{
  "code": 0,
  "msg": "success",
  "data": {
    "deviceId": 123,
    "expireAt": "2026-07-24 14:00:00",
    "status": "ACTIVATING"
  }
}
```

幂等说明：同一设备重复调用时不会消耗新天卡，直接返回已有天卡信息。若天卡已激活成功，status 返回 `ACTIVE`。

错误响应：

code	msg	说明
400	mac/wifiMac/deviceUid/deviceSn 至少传一个	缺少设备标识
400	天卡池库存不足	可用天卡已耗尽
400	操作太频繁，请稍后再试	同一设备短时间内重复请求
500	天卡分配失败，请重试	系统繁忙，请重试

3.2 查询天卡库存

```
GET /openapi/partner/card/stock
```

响应示例：

```
{
  "code": 0,
  "msg": "success",
  "data": {
    "available": 50
  }
}
```

---

4. 媒资接口

4.1 查看授权范围

```
GET /openapi/partner/media/package
```

响应：

字段	说明
scope	授权级别：CONTENT（指定内容）/ CATALOG（指定栏目）/ FULL（全量）
catalogIds	授权栏目 ID（逗号分隔，CATALOG 级别有值）
contentCount	授权内容数量

4.2 栏目树

```
GET /openapi/partner/media/catalog-tree
```

按授权范围过滤返回： – CONTENT 级别返回空树 – CATALOG 级别只返回授权栏目 – FULL 级别返回完整栏目树

4.3 内容列表

```
GET /openapi/partner/media/list?catalogid=472&pageindex=1&pagesize=20
```

参数	类型	必填	说明
catalogid	String	FULL/CATALOG 级别必填	栏目 ID
pageindex	int	否（默认 1）	页码
pagesize	int	否（默认 20）	每页条数

- CONTENT 级别：返回授权内容列表，不需传 catalogid
- CATALOG 级别：必须传授权范围内的 catalogid
- FULL 级别：必须传 catalogid

4.4 搜索

```
GET /openapi/partner/media/search?keyword=镖人&pageindex=1&pagesize=20
```

参数	类型	必填	说明
keyword	String	是	搜索关键词
pageindex	int	否（默认 1）	页码
pagesize	int	否（默认 20）	每页条数

4.5 内容字段说明

字段	说明
contentcode	内容唯一编码
name	内容名称
posterimg	海报图片 URL
categoryname	分类名称
score	评分
isFree	1 = 免费，0 = 付费
episodecount	集数
duration	时长（秒）

字段	说明
releaseyear	上映年份
originalcountry	地区

---

5. APK 侧流程说明

以下接口由 APK 调用，第三方无需直接调用，但需了解流程以便 EPG 传参启动 APK。

5.1 设备初始化

APK 使用消耗接口传入的相同设备标识调用初始化：

```
POST /api/jst/init
Content-Type: application/json
```

请求体（不传 activationCode，代销模式）：

```
{
  "mac": "AA:BB:CC:DD:EE:01",
  "wifiMac": "...",
  "deviceSn": "SN-001",
  "deviceUid": "UID-001"
}
```

返回 `sign`（设备鉴权凭证，24 小时有效）。

5.2 查询天卡权益

APK 轮询查询激活状态，仅当 status 为 ACTIVE 时获取播放凭证：

```
POST /api/jst/checkBenefits
Header: sign: <init 返回的 sign>
```

响应：

status	说明	是否返回 thirdCardNo
ACTIVATING	异步激活中，需继续轮询	否
ACTIVE	激活成功，可播放	是
ACTIVATION_FAILED	激活失败	否
UNBOUND	无天卡绑定	否

轮询建议：间隔 2 秒，超时 5 分钟。

5.3 播放

APK 使用 `thirdCardNo` 作为播放凭证初始化橘视厅 SDK，获取播放地址后播放。

---

6. 签名示例

Java

```
String appKey = "平台分配的appKey";
String appSecret = "平台分配的appSecret";
String timestamp = String.valueOf(System.currentTimeMillis());
String nonce = UUID.randomUUID().toString().replace("-", "");
String sign = DigestUtils.md5Hex(appKey + timestamp + nonce +
appSecret).toUpperCase();
```

Python

```
import hashlib, time, uuid

timestamp = str(int(time.time() * 1000))
nonce = uuid.uuid4().hex
sign = hashlib.md5((app_key + timestamp + nonce +
app_secret).encode()).hexdigest().upper()
```

Node.js

```
const crypto = require('crypto');
const timestamp = Date.now().toString();
const nonce = crypto.randomBytes(16).toString('hex');
const sign = crypto.createHash('md5')
    .update(appKey + timestamp + nonce + appSecret)
    .digest('hex').toUpperCase();
```

---

7. 请求示例（curl）

以下示例中的 `appKey`、`appSecret` 需替换为平台分配的实际值。`timestamp`、`nonce`、`sign` 需按签名算法实时生成。

7.1 消耗天卡

```
curl -X POST "http://58.220.83.58:8084/openapi/partner/card/consume" \
  -H "Content-Type: application/json" \
  -H "appKey: 你的appKey" \
  -H "timestamp: 1784809605695" \
  -H "nonce: 1784809605695aaf094ef" \
  -H "sign: A6EBB96858ED2FAD864CB39568E3EABB" \
  -d '{"mac":"AA:BB:CC:DD:EE:01","deviceSn":"SN-001","deviceUid":"UID-001"}'
```

响应：

```
{
  "code": 0,
  "msg": "success",
  "data": {
    "deviceId": 123,
    "expireAt": "2026-07-24 20:26:46",
    "status": "ACTIVATING"
  }
}
```

7.2 查询天卡库存

```
curl -X GET "http://58.220.83.58:8084/openapi/partner/card/stock" \
  -H "appKey: 你的appKey" \
  -H "timestamp: 1784809605695" \
  -H "nonce: 1784809605695aaf094ef" \
  -H "sign: A6EBB96858ED2FAD864CB39568E3EABB"
```

7.3 查看授权范围

```
curl -X GET "http://58.220.83.58:8084/openapi/partner/media/package" \
  -H "appKey: 你的appKey" \
  -H "timestamp: 1784809605695" \
  -H "nonce: 1784809605695aaf094ef" \
  -H "sign: A6EBB96858ED2FAD864CB39568E3EABB"
```

7.4 栏目树

```
curl -X GET "http://58.220.83.58:8084/openapi/partner/media/catalog-tree" \
  -H "appKey: 你的appKey" \
  -H "timestamp: 1784809605695" \
  -H "nonce: 1784809605695aaf094ef" \
  -H "sign: A6EBB96858ED2FAD864CB39568E3EABB"
```

7.5 内容列表

```
curl -X GET "http://58.220.83.58:8084/openapi/partner/media/list?
catalogid=472&pageindex=1&pagesize=20" \
  -H "appKey: 你的appKey" \
  -H "timestamp: 1784809605695" \
  -H "nonce: 1784809605695aaf094ef" \
  -H "sign: A6EBB96858ED2FAD864CB39568E3EABB"
```

7.6 搜索

```
curl -X GET "http://58.220.83.58:8084/openapi/partner/media/search?keyword=镖人
&pageindex=1&pagesize=20" \
  -H "appKey: 你的appKey" \
  -H "timestamp: 1784809605695" \
```

```
  -H "nonce: 1784809605695aaf094ef" \
  -H "sign: A6EBB96858ED2FAD864CB39568E3EABB"
```

7.7 签名生成示例（bash）

```
APP_KEY="你的appKey"
APP_SECRET="你的appSecret"
TIMESTAMP=$(date +%s%3N)
NONCE="${TIMESTAMP}$(head -c 8 /dev/urandom | xxd -p)"
SIGN=$(echo -n "${APP_KEY}${TIMESTAMP}${NONCE}${APP_SECRET}" | md5sum | awk
'{print toupper($0)}')

echo "appKey: $APP_KEY"
echo "timestamp: $TIMESTAMP"
echo "nonce: $NONCE"
echo "sign: $SIGN"

# 直接用变量发请求
curl -X POST "http://58.220.83.58:8084/openapi/partner/card/consume" \
  -H "Content-Type: application/json" \
  -H "appKey: $APP_KEY" \
  -H "timestamp: $TIMESTAMP" \
  -H "nonce: $NONCE" \
  -H "sign: $SIGN" \
  -d '{"mac":"AA:BB:CC:DD:EE:01","deviceSn":"SN-001"}'
```

---

8. 注意事项

1. **nonce 唯一性**：每个请求的 nonce 必须唯一，5 分钟内重复提交会被拒绝。
2. **timestamp 有效期**：与服务器时间差超过 5 分钟会过期。
3. **设备标识一致性**：消耗接口传入的设备标识必须与 APK 启动时传入的一致，否则 APK 无法找到设备。
4. **异步激活**：消耗接口返回后天卡异步激活（通常 2–5 秒），APK 需轮询 checkBenefits 确认激活完成。
5. **库存管理**：消耗接口返回库存不足错误时，需联系平台采购天卡。

附录：设备标识说明

deviceUid 生成规则

APK 的 `deviceUid` 按以下优先级生成：

优先级	来源	格式	重装不变	工厂重置不变
1	Android ID（Settings.Secure.ANDROID_ID）	64 位 hex（16 字符），如 `f0f179fe199f7a52`	是	否
2	UUID（兜底，存 SharedPreferences）	32 位无横线，如 `a1b2c3d4e5f6...`	否	否

Android ID 为空时使用 UUID 兜底。正常情况下 deviceUid 即 Android ID。

关键注意点

1. **consume 与 init 的 deviceUid 必须一致**：第三方调用消耗接口时传入的 `deviceUid` 必须与 APK 调用 init 时传入的 `deviceUid` 相同，否则 APK 无法找到设备。
2. **Android ID 按签名隔离**：Android 8+ 系统中，不同签名的 App 在同一设备上获取的 Android ID 不同。因此 EPG 获取 deviceUid 的方式需与 APK 一致，建议由 APK 提供给 EPG，而非 EPG 自行获取。
3. **推荐传参方式**：EPG 启动 APK 时通过 Intent 传递设备标识，APK 使用相同标识调 init 接口。设备标识建议优先使用 `mac`，其次 `deviceUid`。

设备标识匹配优先级

消耗接口和 init 接口均按以下优先级匹配设备：

```
mac > wifiMac > deviceUid > deviceSn
```

只要有一个标识匹配即可找到设备。建议 consume 和 init 传递相同的标识组合。