微信小程序后台播放接入说明

本文介绍保利威直播在微信小程序中的后台小窗播放对接方式，覆盖以下两种观看场景：

- 原生小程序观看页：由观看页播放器直接请求后台小窗播放。
- 小程序 WebView 观看页：H5 观看页跳转到小程序原生中转页，由中转页创建原生 `<video>` 并请求后台小窗播放。

后台小窗始终由微信小程序原生 `<video>` 或 `<live-player>` 组件承载。`requestBackgroundPlayback()` 不接收播放地址，直播流或回放地址必须先绑定到当前正在播放的原生组件，再通过对应的 Context 请求后台播放。

微信小程序核心 API

微信小程序实现后台播放的核心是播放器 Context 上的 `requestBackgroundPlayback()`。Polyv SDK 提供的 `watchCore.player.requestBackgroundPlayback()` 是对微信原生 Context API 的统一封装，不是另一套系统小窗能力。

进入后台小窗

微信原生 API	官方说明	最低基础库	适用组件
`VideoContext.requestBackgroundPlayback()`	进入后台小窗播放模式	`2.14.3`	`<video>`
`LivePlayerContext.requestBackgroundPlayback()`	进入后台小窗播放模式	`2.14.3`	`<live-player>`

两个 API 均无参数、返回 `void`。使用方式如下：

```
// video
const videoContext = wx.createVideoContext('videoId', this);
videoContext.requestBackgroundPlayback();

// live-player
const livePlayerContext = wx.createLivePlayerContext('livePlayerId', this);
livePlayerContext.requestBackgroundPlayback();
```

Context 的创建方式可参考微信官方文档：

- `wx.createVideoContext()`
- `wx.createLivePlayerContext()`

在 Polyv 原生观看页中，业务代码只需要调用：

```
watchCore.player.requestBackgroundPlayback();
```

SDK 会根据当前实际播放节点，向下转发到 `VideoContext.requestBackgroundPlayback()` 或 `LivePlayerContext.requestBackgroundPlayback()`。

退出后台小窗

微信原生 API	官方说明	最低基础库	适用组件
`VideoContext.exitBackgroundPlayback()`	退出后台小窗播放模式	`2.14.3`	`<video>`
`LivePlayerContext.exitBackgroundPlayback()`	退出后台小窗播放模式	`2.14.3`	`<live-player>`

WebView 中转页返回 H5 前使用的是 `VideoContext.exitBackgroundPlayback()`：

```
const videoContext = wx.createVideoContext('polyvLiveVideo', this);
videoContext.exitBackgroundPlayback();
wx.navigateBack({ delta: 1 });
```

原生控制栏按钮

`video` 组件的 `show-background-playback-button` 用于控制原生控制栏是否展示后台播放按钮。该属性只控制按钮展示，不代替 Context API；使用自定义按钮时仍需主动调用 `requestBackgroundPlayback()`。

接入前准备

接入前请确认：

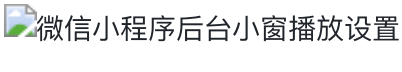
- 已在保利威直播管理后台开启对应的小窗播放设置。
- 微信小程序基础库需要为 `2.14.3` 或以上，低版本必须做兼容处理。

- 使用 iOS、Android 真机验证。开发者工具不能替代应用切后台场景的真机结果。
- 调用时播放器已经完成初始化并进入播放状态，原生播放器节点仍然挂载在页面中。
- 当前微信版本和操作系统支持后台小窗播放。

该方案不依赖 `app.json` 的 `requiredBackgroundModes`。`requiredBackgroundModes` 中的 `audio` 是后台音频能力，不是视频后台小窗开关。

管理后台设置

在直播管理后台的频道小程序设置中，可以看到以下配置：



设置项	作用	当前适用范围
小程序原生小窗播放	控制原生小程序观看页是否展示小窗入口	直播、回放
小程序 WebView 小窗播放	控制 WebView H5 观看页是否展示小窗入口	直播
小程序原生中间页路径	WebView 观看页点击小窗后跳转的原生小程序页面路径	仅 WebView 方案

中转页路径

当前 Polyv 观看端提供的参考中转页是：

```
/pages-other/pages/window/window
```

实现可参考 Polyv 观看端源码中的：

```
src/pages-other/pages/window/
```

中转页必须在小程序 `app.json` 的 `pages` 或分包页面中完成注册。WebView 观看页必须使用 `wx.miniProgram.navigateTo` 打开该页面，以保留上一层 WebView 页面；不要使用 `redirectTo` 替换 WebView 页面，否则中转页无法通过 `navigateBack` 返回原观看页。

截图中的配置示例为：

```
/pages-other/pages/window/window?appId=xxxx&appSecret=xxxx&accountId=xxxx
```

当前参考页实际需要以下四个参数：

参数	必填	说明
`channelId`	是	当前直播频道 ID，通常由 H5 观看页在点击时追加
`accountId`	是	保利威帐号 ID，不是观众 `userId`
`appId`	是	当前验证实现使用的播放器鉴权 App ID
`appSecret`	是	仅当前验证实现用于前端计算签名

参数名区分大小写。`accountId` 表示保利威帐号 ID，观众 `userId` 表示观看用户标识，二者是完全不同的业务字段，不能互相替代，也不能把观众 `userId` 作为 `accountId` 传入。所有动态参数都应执行 `encodeURIComponent`。

> 安全提示：截图展示的是当前验证环境的配置方式。生产环境禁止把长期 `appSecret` 放在管理后台页面路径、H5 JavaScript、小程序路由参数或小程序代码中。正式接入应由业务服务端保管密钥，只向前端签发短时、最小权限的播放凭证。中转页可接收一次性 `ticket`，再向业务服务端换取短时 `sign + timestamp` 或播放 token。采用该方式前，需要同步改造当前 Polyv `window` 参考页的鉴权入参。

原生小程序观看页实现

实现流程

当前 Polyv 观看端只在竖屏观看页展示小窗按钮。按钮位于频道信息胶囊旁，显示条件为：

- 后台已开启“小程序原生小窗播放”。
- 当前频道状态为直播中或回放中。

调用链如下：

```
sequenceDiagram
  participant User as 用户
  participant Page as 原生观看页
  participant SDK as watchCore.player
  participant Controller as 播放器控制器
  participant Context as VideoContext 或 LivePlayerContext
  participant System as 微信或系统小窗
  User->>Page: 点击小窗按钮
  Page->>SDK: requestBackgroundPlayback()
  SDK->>Controller: 转发到当前直播或点播控制器
  Controller->>Context: requestBackgroundPlayback()
  Context->>System: 请求进入后台小窗
```

对应的核心调用是：

```
watchCore.player.requestBackgroundPlayback();
```

SDK 会根据当前播放器类型转发调用：

- 直播播放器：获取当前播放器组件暴露的 `getVideoContext()`，最终调用 `VideoContext` 或 `LivePlayerContext`。
- 点播播放器：调用点播播放器上下文的 `requestBackgroundPlayback()`。

当前本地播放器会根据媒体类型选择原生节点：

- 普通直播流优先使用 `<live-player>`。
- `.m3u8`、回放、暖场视频或开启 `forceVideo` 时使用 `<video>`。

推荐调用时机

应在播放器真正开始播放后开放小窗按钮。观看页 SDK 可以监听 `PlayerEvents.PlayerPlaying`：

```
import { PlayerEvents } from '@polyv/live-watch-miniprogram-sdk';

let canRequestBackgroundPlayback = false;

watchCore.player.eventEmitter.on(PlayerEvents.PlayerPlaying, () => {
  canRequestBackgroundPlayback = true;
});

function onTapBackgroundPlayback() {
  if (!canRequestBackgroundPlayback) {
    wx.showToast({
      title: '请先开始播放',
      icon: 'none',
    });
    return;
  }

  try {
    watchCore.player.requestBackgroundPlayback();
  } catch (error) {
    wx.showToast({
      title: '小窗打开失败',
      icon: 'none',
    });
  }
}
```

`requestBackgroundPlayback()` 无参数、返回 `void`。调用没有返回成功回调，因此不能把“方法未抛错”直接等同于“系统小窗已经打开”。

直接使用原生 video

如果没有使用观看页 SDK，也可以直接对原生 `<video>` 创建 Context：

```
<video
  id="liveVideo"
  src="{{ videoSrc }}"
  autoplay
  show-background-playback-button="{{ true }}"
  bindplay="onVideoPlay"
/>

<button bind:tap="onTapBackgroundPlayback">小窗播放</button>
```

```
Page({
  data: {
    isPlaying: false,
  },

  onVideoPlay() {
    this.setData({ isPlaying: true });
  },

  onTapBackgroundPlayback() {
    if (!this.data.isPlaying) {
      wx.showToast({ title: '请先开始播放', icon: 'none' });
      return;
    }

    const videoContext = wx.createVideoContext('liveVideo', this);
    if (typeof videoContext.requestBackgroundPlayback !== 'function') {
      wx.showToast({ title: '当前环境不支持小窗播放', icon: 'none' });
      return;
    }

    try {
      videoContext.requestBackgroundPlayback();
    } catch (error) {
      wx.showToast({ title: '小窗打开失败', icon: 'none' });
    }
  },
});
```

`show-background-playback-button` 只控制原生控制栏是否显示后台播放按钮；使用自定义按钮时仍需主动调用 Context API。

页面生命周期

系统小窗依赖原播放器页面和原生节点存活。后台播放期间不要执行以下操作：

- `navigateBack`、`redirectTo` 或其他会卸载播放器页面的路由操作。
- 销毁观看页 SDK 或播放器实例。
- 通过 `wx:if` 移除 `<video>` / `<live-player>`。
- 清空 `src` 或切换到另一个播放器 Context。

当前 Polyv 原生观看页针对 iOS 返回页面后的恢复问题做了兼容处理：触发过后台播放后，页面再次显示会立即调用一次 `play()`，并在 1 秒后再次调用。该处理可以缓解 `<video>` 播放 m3u8 时第一次 `play()` 无效的问题；当前 `<live-player>` 在系统小窗中暂停后返回，仍可能无法通过 `play()` 恢复，需要结合真机结果继续处理。

WebView 观看页实现

方案定位

WebView 页面不能直接获取小程序原生 `VideoContext`。当前方案通过一个可见的原生中转页接管播放：

```
sequenceDiagram
  participant User as 用户
  participant H5 as WebView 观看页
  participant Window as 原生 window 中转页
  participant Core as wx-live-player-core
  participant Video as 原生 video
  participant System as 微信或系统小窗
  User->>H5: 点击小窗播放
  H5->>Window: wx.miniProgram.navigateTo
  Window->>Core: 创建播放器并请求媒体信息
  Core-->>Window: 返回 mediaInfo.src
  Window->>Video: 设置 src 并自动播放
  Video-->>Window: bindplay
  Window->>System: requestBackgroundPlayback()
  System-->>Window: 用户返回微信
  Window->>Video: exitBackgroundPlayback()
  Window->>H5: navigateBack 返回 WebView
```

当前完整流程是：

1. 小程序登录页通过 `redirectTo` 打开承载 H5 观看页的 WebView 页面。
2. H5 根据后台的“小程序 WebView 小窗播放”开关决定是否展示小窗按钮。
3. 用户点击后，H5 暂停自身播放器，并使用 `wx.miniProgram.navigateTo` 打开后台配置的原生中转页。
4. 中转页获取直播流地址，挂载原生 `<video>`。
5. `<video>` 触发 `bindplay` 后，中转页调用 `requestBackgroundPlayback()`。
6. 用户从系统小窗返回微信时，中转页在后续 `onShow` 中退出后台播放，再 `navigateBack` 返回原 WebView 页面。
7. 中转页 `onUnload` 时销毁播放器。

因此该方案的播放归属会发生切换：进入中转页后由原生播放器播放；返回 WebView 前会结束原生系统小窗并销毁中转播放器。它不支持“已经返回 WebView 页面，但原生系统小窗仍继续播放”的并行状态。

H5 跳转中转页

WebView 网页需要引入微信 JSSDK，并在用户点击时调用：

```
<script src="https://res.wx.qq.com/open/js/jweixin-1.3.2.js"></script>
```

```
function openBackgroundPlayback(options) {
  const intermediatePagePath = options.intermediatePagePath;
  const channelId = options.channelId;
  const separator = intermediatePagePath.includes('?') ? '&' : '?';
  const targetUrl =
    intermediatePagePath +
    separator +
    'channelId=' +
    encodeURIComponent(channelId);

  // 跳转前暂停 H5 播放器，避免双音频和重复统计。
  options.pauseWebPlayer();

  wx.miniProgram.navigateTo({
    url: targetUrl,
    fail(error) {
      console.error('打开小窗中转页失败', error);
      options.resumeWebPlayer();
    },
  });
}
```

`intermediatePagePath` 使用管理后台返回的中转页路径。当前截图中的路径已经包含 `appId`、`appSecret` 和 `accountId`，H5 在点击时追加当前 `channelId`。正式环境应改为追加短时 `ticket`，不要传递长期 `appSecret`。

Polyv window 中转页逻辑

Polyv 观看端的 `pages-other/pages/window/window` 当前使用原生 `Page({...})` 生命周期，核心逻辑如下：

1. 解析并校验 `channelId`、`accountId`、`appId`、`appSecret`。
2. 生成时间戳和 MD5 签名，创建 `@polyv/wx-live-player-core` 播放器。
3. 设置 `forceVideo: true`，确保直播由原生 `<video>` 承载。
4. 监听 `UPDATE_MEDIA_INFO`，把 `mediaInfo.src` 写入页面的 `videoSrc`。
5. 原生 `<video>` 起播后创建 `VideoContext`，并且只发起一次 `requestBackgroundPlayback()`。
6. 后续 `onShow` 检查上一页是否为 WebView；满足条件时先退出后台播放，再返回 WebView。
7. `onUnload` 销毁 `PolyvLive`，避免播放器和事件监听残留。

参考页会把 `accountId` 传给 `PolyvLive` 的 `uid` 配置，并用于 `statistics.param1`。这里的 `accountId` 仍然是保利威帐号 ID，不代表观众 `userId`。

中转页的原生节点应保持可见和挂载：

```
<video
  wx:if="{{ videoSrc }}"
  id="polyvLiveVideo"
  src="{{ videoSrc }}"
  autoplay
  controls="{{ false }}"
  enable-progress-gesture="{{ false }}"
  show-fullscreen-btn="{{ false }}"
  object-fit="contain"
  bindplay="onVideoPlay"
  binderror="onVideoError"
/>
```

不要对播放器使用 `display: none`、`visibility: hidden`、`opacity: 0`，也不要在发起后台播放后立即卸载中转页。

生产鉴权改造

当前参考页把 `appSecret` 带到前端并在小程序中计算签名，只能用于内部验证。推荐的生产流程是：

1. H5 向业务服务端申请一次性、短有效期的 `ticket`。
2. H5 只把 `channelId` 和 `ticket` 传给原生中转页；如果业务还需要观众 `userId`，应使用独立字段传递，不能覆盖 `accountId`。
3. 中转页用 `ticket` 向业务服务端换取短时播放器鉴权信息。
4. 业务服务端持有 `appSecret` 并完成签名，前端永远不接触长期密钥。
5. 中转页使用返回的短时 `appId + sign + timestamp` 或播放 token 创建播放器。

`ticket` 应绑定频道、观众、过期时间和使用次数，并在服务端校验，避免被复制到其他频道或重复使用。

用户交互与调用时机

微信官方对 `requestBackgroundPlayback()` 的接口说明没有标注“只能由用户点击调用”，但这不等于进入页面后可以无条件立即调用：

- 原生媒体可能尚未创建 Context 或尚未开始播放。
- 自动播放可能受终端、系统或媒体策略影响。
- 后台小窗是否出现仍由微信和操作系统共同决定。

推荐做法：

- 原生观看页由用户点击小窗按钮触发。

- WebView 方案由用户点击 H5 小窗按钮进入中转页，并等原生 `<video>` 的 `bindplay` 后调用。
- 不要在 `onLoad` 中创建 Context 后立即调用。
- 不要在调用后紧接着执行 `wx.exitMiniProgram()`。微信没有保证这组调用的组合时序。

状态判断与错误处理

建议同时使用以下方式维护状态：

- 调用前检查 `requestBackgroundPlayback` 是否存在，并捕获同步异常。
- 提供手动重试入口，不要只依赖自动调用一次。
- 在页面生命周期中记录是否已经发起后台播放请求，避免重复调用。

当前 Polyv `window` 参考页中的 `backgroundPlaybackRequested` 只表示“已经发起调用且未同步抛错”，不代表后台小窗真实进入。业务侧需要保留失败提示和手动重试入口。

后台播放限制

- 微信没有提供直接把流地址传给后台小窗的全局 API。
- 必须先让微信原生 `<video>` 或 `<live-player>` 播放，再调用对应 Context。
- 强制结束微信进程、卸载播放器页面、销毁播放器或清空媒体源后，后台播放无法继续。
- 用户把微信切到后台后小窗立即消失时，应优先检查基础库版本、播放器是否已起播、原生节点是否仍存活，以及当前微信和操作系统是否支持该能力。
- 不存在额外的小程序参数可以强制后台小窗继续显示。

当前参考项目注意事项

- 原生观看页按钮目前只判断后台开关和频道状态，没有判断播放器是否已真正起播。业务接入时建议增加 `PlayerEvents.PlayerPlaying` 状态保护。
- WebView 中转页通过 `backgroundPlaybackRequested` 记录是否发起调用，但该状态不能确认后台小窗是否实际打开。
- WebView 中转页只用于直播；管理后台截图也明确标注 WebView 小窗仅直播中可用。

常见问题

Q1：点击按钮后为什么没有出现小窗?

A： 请确认播放器已经进入播放状态、基础库版本符合要求、Context 方法存在、原生播放器节点未被卸载，并使用真机测试。

Q2：原生观看页为什么没有小窗按钮?

A： 请确认管理后台已开启“小程序原生小窗播放”，并且频道当前状态为直播中或回放中。

Q3：WebView 点击后为什么无法返回原观看页?

A： 请确认 H5 使用 `wx.miniProgram.navigateTo` 打开中转页，且页面栈中的上一页仍为 `pages-other/pages/web-view/web-view`。若使用 `redirectTo`，原 WebView 页面会被替换。

Q4：WebView 和中转页为什么会同时有声音?

A： H5 跳转前应暂停 Web 播放器；返回 WebView 后，再由 H5 决定是否恢复播放。当前方案不会自动同步 H5 与原生播放器的播放状态。

Q5：iOS 切到微信后台后，为什么小窗也消失了?

A： 请检查基础库版本、播放器是否已经起播、原生节点是否仍然挂载，以及当前微信版本和操作系统是否支持后台小窗。小程序没有额外参数可以强制后台小窗继续显示。

微信官方文档

- video 组件
- live-player 组件
- VideoContext.requestBackgroundPlayback
- VideoContext.exitBackgroundPlayback
- LivePlayerContext.requestBackgroundPlayback
- LivePlayerContext.exitBackgroundPlayback
- wx.createVideoContext
- wx.createLivePlayerContext
- web-view 组件