鸿蒙 点播回放 Core SDK

- 包名：`@baijia/videoplayer`（har 包）
- **OHPM 仓库**：`https://ohpm.openharmony.cn/#/cn/detail/@baijia%2Fvideoplayer`
- Platform：HarmonyOS NEXT，runtimeOS: `HarmonyOS`，compileSdkVersion: `6.0.2(22)`
- IDE：**DevEco Studio 6.0.2 Release**（请通过 GUI 编译部署）
- 底层依赖：`@baijia/ijkplayer` v2.0.6（har 预编译包）

1. 简介

百家云鸿蒙端点播回放 Core SDK 是一个集点播和回放于一体的无 UI 纯实现库，点播功能包括在线视频、本地播放、视频缓存；回放功能在点播播放器基础上叠加 PPT、聊天、答题、画笔、在线人员等模块，支持离线回放，还原直播场景。

> *鸿蒙端实现逐行对齐 Android `videoplayer-core` 接口，并做了平台适配：*
>
> - *Android `bindPlayerView(BJYPlayerView)` → 鸿蒙端 `setContext(XComponent context, string id)`（直接绑定 `XComponent` 的 surface）*
> - *Android `RxJava Observable` → 鸿蒙端 `Promise` 或自定义 `LPEventEmitter<T>`*
> - *Android `BJYPlayerSDK.Builder` 链式初始化 → 鸿蒙端在 Core 层提供 `BJYPlayerConfig` 单例配置*

> *（UI 层再封装 `BJYPlayerSDKBuilder`，见 UI 文档 §2.4）*

1.1. 功能描述

功能	描述
在线播放	支持百家云后台配置视频播放（鉴权、清晰度、CDN）
播放器视图	通过 `XComponent` 承载渲染，无 UI 控件
视频缓存	LRU 缓存，基于 `relationalStore` 持久化（`VideoCacheManager`）
离线播放	支持下载后的本地视频/回放，加密/不加密均支持
回放	PPT、聊天、答题、画笔、在线人员、公告、红包等信令模块
下载	点播下载、回放下载，断点续传，CDN 兜底

2. 快速集成

2.1. 包依赖

`oh-package.json5` 引入：

```
{
 "dependencies": {
   "@baijia/videoplayer": "^1.0.0"
 }
}
```

> `@baijia/videoplayer` 已传递依赖 `@baijia/ijkplayer`，无需额外声明。

2.2. 初始化 SDK

通过 `BJYPlayerConfig` 单例配置全局参数（鸿蒙端在 Core 层等价于 Android `BJYPlayerSDK.Builder`）。

```
import { BJYPlayerConfig, DeployType } from '@baijia/videoplayer';

const config = BJYPlayerConfig.getInstance();
config.customDomain = 'demo123';       // 专属域名前缀
config.isEncrypt = true;               // 启用加密
config.isDevelopMode = false;          // 正式发布请置 false
config.deployType = DeployType.PRODUCTION;
```

`BJYPlayerConfig` 全部可配置项（`videoplayer/src/main/ets/config/BJYPlayerConfig.ets:25`）：

字段	类型	默认值
`customDomain`	`string`	`''`
`environmentInfix`	`string`	`'at'`
`environmentSuffix`	`string`	`'baijiayun.con`
`apiPrefix`	`string`	`'www'`
`deployType`	`DeployType`	`PRODUCTION`

isEncrypt	boolean	false
isDevelopMode	boolean	false

工具方法：

```
config.getApiBaseUrl(): string;     // 根据 customDomain + deployType 拼接 API host
config.getClickUrl(): string;       // 统计上报域名
```

> *Android 端 `BJYPlayerSDK.STATIC_PPT_DEFAULT_SIZE` 、 `DISABLE_ANIM_PPT` 、 `waterMark` 等全局静态字段，在鸿蒙端归到 UI 层 `BJYPlayerSDK` 类（详见 UI 文档 §2.4）。Core 层只保留与播放器引擎相关的全局配置。*

3. 点播部分

3.1. 播放视频

3.1.1. 创建播放器

通过 `VideoPlayerFactory` 创建播放器（ `videoplayer/src/main/ets/player/VideoPlayerFactory.ets:12` ）。

```
import { VideoPlayerFactory, IBJYVideoPlayer } from '@baijia/videoplayer';

```

```
const videoPlayer: IBJYVideoPlayer =
VideoPlayerFactory.createPlayer();
```

API	用途
`static createPlayer(): IBJYVideoPlayer`	创建标准点播/回放播放器
`static createMixedPlayer(): BJYVideoPlayerMixedImpl`	创建合并回放专用播放器（对 `Builder.setMixedPlayback`

> *Android 的 `VideoPlayerFactory.Builder` 链式参数（`setSupportLooping` / `setSupportBackgroundAudio` / `setSupportBreakPointPlay` / `setLifecycle` 等）在鸿蒙端**未实现 Builder 模式**，请通过 `IBJYVideoPlayer` 实例上的 setter 在创建后设置（见 §3.2.1）。*

3.1.2. 视频渲染视图（XComponent）

鸿蒙端**不存在** `BJYPlayerView` 组件，统一使用 `XComponent` 承载视频渲染。声明 `XComponent` 时必须指定 `libraryname: 'ijkplayer_napi'`，并把 `XComponentController` 上下文和 id 通过 `setContext(context, id)` 交给播放器（对应 Android `bindPlayerView`）。

```
@Entry
@Component
struct PlayerPage {
  private xComponentController:
XComponentController = new
XComponentController();
```

```
  private xComponentId: string = 'videoXComponent';
  private videoPlayer: IBJYVideoPlayer = VideoPlayerFactory.createPlayer();

  build() {
    Stack() {
      XComponent({
        id: this.xComponentId,
        type: XComponentType.SURFACE,
        libraryname: 'ijkplayer_napi',
        controller: this.xComponentController
      })
        .onLoad((event?: object) => {
          // event 由 ijkplayer_napi 注入，包含 native context
          this.videoPlayer.setContext(event as object, this.xComponentId);
        })
        .width('100%')
        .height(200)
    }
  }
}
```

> *视频画面裁剪/比例由 `XComponent` 的尺寸约束自行实现，Core SDK 不再提供 `AspectRatio` 枚举。*

3.1.3. 绑定视频源

`IBJYVideoPlayer` 提供 3 个数据源入口（`videoplayer/src/main/ets/player/IBJYVideoPlayer.ets`）：

```
// 在线视频（视频 ID + token，accessKey 可传空串）
setupOnlineVideoWithId(videoId: number, token: string, accessKey: string): void;   // line 41

// 在线视频（已构建好的 VideoItem）
setupOnlineVideoWithVideoItem(videoItem: VideoItem): void;                          // line 47

// 本地视频（DownloadManager 下载后的 DownloadModel）
setupLocalVideoWithDownloadModel(downloadMod object): void;                        // line 35
```

> *设置完视频源后是否自动播放由 `setAutoPlay(boolean)` 决定，默认自动播放。*

3.2. 设置播放器

3.2.1. 设置参数

`IBJYVideoPlayer` 提供的 setter API（`videoplayer/src/main/ets/player/IBJYVideoPlayer.ets`）：

API	说明
`setUserInfo(userName: string, userIdentity: string): void`	设置第三方用户信息，用于后台统计
`setUserGroup(group: number): void`	设置用户分组
`supportBackgroundAudio(enable: boolean): void`	是否后台播放音频

`supportLooping(looping: boolean): void`	是否循环播放
`setAutoPlay(autoPlay: boolean): void`	setup 后是否自动播放
`enableBreakPointMemory(helper: BreakPointMemoryHelper, videoId: string): void`	启用断点续播
`setPlayRate(rate: number): void`	倍速播放 `[0.5 ~ 2.0]`

3.2.2. 获取播放器状态

```
getCurrentPosition(): number;        // 当前位置（秒）              line 76
getDuration(): number;               // 视频总时长（秒）            line 79
getBufferPercentage(): number;       // 缓冲百分比                  line 82
getPlayRate(): number;               // 当前倍速                    line 88
isPlaying(): boolean;                // 是否正在播放                line 73
isPlayLocalVideo(): boolean;         // 是否播放本地视频            line 94
getPlayerStatus(): PlayerStatus;     // 播放状态枚举（见 §6.1）     line 85
getVideoInfo(): BJYVideoInfo | null; // 视频信息（见 §6.3）         line 91
getVideoMemoryPoint(): number;       // 记忆播放位置                line 100
getMediaPlayerDebugInfo(): object;   // 调试信息                    line 97
```

```
getVideoWidth(): number;          // 解码视频宽度（像素）        line 205
getVideoHeight(): number;         // 解码视频高度（像素）        line 208
getBreakPoint(): number;          // 断点位置（毫秒），无则 0    line 155
```

3.2.3. 视频控制

```
play(): void;                                  // line 50
playFromOffset(startOffset: number): void;     // 从指定秒数开始播放（对齐 Android play(int startOffset)） line 53
rePlay(): void;                                // line 56
pause(): void;                                 // line 59
stop(): void;                                  // line 62
seekTo(timeSec: number): void;                 // seek 到指定秒数（对齐 Android seek(int)） line 65
release(): void;                               // line 68
```

清晰度与 CDN：

```
setPreferredDefinitions(definitions: VideoDefinition[]): void;  // 清晰度偏好（按数组顺序优先匹配） line 114
changeDefinition(definition: VideoDefinition): boolean;         // 切换清晰度，播放中调用          line 108
getCDNCount(): number;                                          // CDN 线路数量                    line 117
setCDNIndex(index: number): void;                               // 切换 CDN 线路                   line 120
getCDNIndex(): number;                                          // 当前 CDN 线路 index             line 123
```

清晰度优先级示例：

```
import { VideoDefinition } from '@baijia/videoplayer';

const preferred: VideoDefinition[] = [
  VideoDefinition._720P,
  VideoDefinition.SHD,
  VideoDefinition.HD,
  VideoDefinition.SD,
  VideoDefinition._1080P,
  VideoDefinition.Audio,
];
videoPlayer.setPreferredDefinitions(preferred);
```

3.3. 设置监听

3.3.1. 播放器状态

```
import { OnPlayerStatusChangeListener, PlayerStatus } from '@baijia/videoplayer';

class MyStatusListener implements OnPlayerStatusChangeListener {
  onStatusChange(status: PlayerStatus): void {
    if (status === PlayerStatus.STATE_PREPARED) {
      // 数据已准备好
    }
  }
}
videoPlayer.addOnPlayerStatusChangeListener(new MyStatusListener());
```

> ⚠ *ArkTS 严格模式不允许匿名对象字面量实现接口，**必须用具名 `class`**。*

`IBJYVideoPlayer` 监听器汇总（`IBJYVideoPlayer.ets`）：

add / remove	监听接口
`addOnPlayerStatusChangeListener` / `removeOnPlayerStatusChangeListener`	`OnPlayerSt`
`addOnPlayingTimeChangeListener` / `removeOnPlayingTimeChangeListener`	`OnPlayingT`
`addOnBufferingListener` / `removeOnBufferingListener`	`OnBufferin`
`addOnPlayerErrorListener` / `removeOnPlayerErrorListener`	`OnPlayerE`
`addOnPlayerLagListener` / `removeOnPlayerLagListener`	`OnPlayerE`
`addOnSeekCompleteListener` / `removeOnSeekCompleteListener`	`OnSeekCo`
`addOnVideoSizeChangeListener`	`OnVideoSi`
`addOnFirstFrameListener` / `removeOnFirstFrameListener`	`OnFirstFra`
`addOnVideoTypeChangeListener` / `removeOnVideoTypeChangeListener`	`OnVideoTy`
`addOnSeekSwitchVideoListener`	`OnSeekSw`
`setOnTokenInvalidListener`	`OnTokenIn`

3.3.2. 播放进度

```
import { OnPlayingTimeChangeListener } from '@baijia/videoplayer';

class MyTimeListener implements OnPlayingTimeChangeListener {
  // currentTime: 当前位置（秒）
  // duration:    总时长（秒）
  onPlayingTimeChange(currentTime: number, duration: number): void { }
}
videoPlayer.addOnPlayingTimeChangeListener(new MyTimeListener());
```

3.3.3. seek 结束

接口签名（ `listeners/OnSeekCompleteListener.ets:5` ）：

```
export interface OnSeekCompleteListener {
  // beforeSeekPosition: seek 前位置（秒）
  // seekPosition:       seek 目标位置（秒）
  onSeekComplete(beforeSeekPosition: number, seekPosition: number): void;
}
```

3.3.4. 播放出错

接口签名（ `listeners/OnPlayerErrorListener.ets:7` ），错误信息以 `LPError` （重导自 `@baijia/livebase` ）描述：

```
export interface OnPlayerErrorListener {
  onError(error: LPError): void;
}
```

3.3.5. token 失效

接口签名（ listeners/OnTokenInvalidListener.ets ）：

```
export interface OnTokenFetchedListener {
  onTokenFetchSuccess(token: string): void;
}

export interface OnTokenInvalidListener {
  onTokenInvalid(vid: string, callback: OnTokenFetchedListener): void;
}
```

```
class MyTokenListener implements OnTokenInvalidListener {
  onTokenInvalid(vid: string, callback: OnTokenFetchedListener): void {
    // 调用集成方业务后端获取新 token
    const newToken = await fetchToken(vid);
    callback.onTokenFetchSuccess(newToken);
  }
}
videoPlayer.setOnTokenInvalidListener(new MyTokenListener());
```

3.4. 其它功能

3.4.1. 字幕

```
addCubChangeListener(listener: OnCubChangeListener): void;          // line 160
toggleSubtitleEngine(shutDown: boolean): void;                      // line 163
changeSubtitlePath(subtitlePath: string): void;                     // 单语    line 166
```

```
changeSubtitlePathBilingual(subtitleZhPath: string, subtitleEnPath: string): void; // 双语 line 169
subtitleDefaultEnabled(): boolean; // 默认是否开启 line 172
```

字幕引擎实现：`subtitle/SubtitleEngine.ets`、`subtitle/WebVttParser.ets`，支持 WebVTT 格式。

3.4.2. 弹题

```
getVideoQuizList(videoId: string, userNumber: string, token: string,
                 listener: OnVideoQuizListUpdateListener): void;     // line 191
sendVideoQuizAnswer(answerMap: Map<string, string>): void;          // line 195
```

3.4.3. 关键帧（书签）

```
requestKeyFrameModel(videoId: string): Promise<KeyFrameModel>;      // line 200
```

3.4.4. 水印

水印通过 `WatermarkOverlay`（`widget/WatermarkView.ets`）ArkUI 组件叠加在 `XComponent` 之上，支持四角定位（`WatermarkPosition`）。

```
import { WatermarkOverlay, WatermarkInfo, WatermarkPosition } from '@baijia/videoplayer';
```

```
2.
3. const info = new WatermarkInfo();
4. info.url = 'https://...';
5. info.position = WatermarkPosition.LEFT_TOP;
```

> *服务端配置的水印通过 `IBJYVideoPlayer.getVideoInfo()` → `VideoItem.waterMark` 透传。*

3.4.5. 视频缓存

`VideoCacheManager`（`cache/VideoCacheManager.ets`）使用 `relationalStore` LRU 缓存视频片段。

> *`relationalStore` 无同步 API（`querySync` 等不存在），调用全部走 `async`。*

3.4.6. EV2 加密

`Ev2Decoder`（`crypto/Ev2Decoder.ets`）解码加密视频 URL，加密视频对应 CDN 的 `enc_url` 字段。

3.4.7. 视频统计上报

`StatisticsReporter`（`network/StatisticsReporter.ets`）按 `reportInterval` 周期向 `getClickUrl()/gs.gif` 上报播放行为，无需手动调用。

4. 回放部分

4.1. 快速集成

4.1.1. 创建播放器

参考 §3.1.1。

4.1.2. 创建 PBRoom

鸿蒙端 PBRoom 实现类 `PBRoomImpl`（`videoplayer/src/main/ets/playback/PlaybackRoom.ets`）通过**静态工厂方法**创建，对齐 Android `BJYPlayerSDK.newPlayBackRoom()` 系列重载：

```
import { PBRoomImpl, PBRoom, DownloadModel } from '@baijia/videoplayer';
```

工厂方法	用途
`static createOnlineSimple(classId: number, token: string): PBRoomImpl`	普通在线回放
`static createOnlineWithSession(classId: number, sessionId: number, token: string): PBRoomImpl`	长期课分段回放
`static createOnline(classId: number, sessionId: number, version: number, token: string): PBRoomImpl`	裁剪回放（version 为裁剪版本）
`static createMixed(mixedId: string, mixedToken: string): PBRoomImpl`	合并回放
`static createOffline(videoDownloadModel: DownloadModel, signalDownloadModel: DownloadModel): PBRoomImpl`	离线回放

示例：

```
const pbRoom: PBRoom = PBRoomImpl.createOnlineWithSession(classId, sessionId, classToken);
```

4.1.3. 绑定播放器

```
pbRoom.bindPlayer(videoPlayer);                    // 主播放器
pbRoom.bindCloudVideoPlayer(secondaryPlayer);      // 可选：云端视频副播放器
```

`PBRoom` 接口（`videoplayer/src/main/ets/playback/PlaybackInterfaces.ets:60`）主要 API：

API	说明
`enterRoom(listener: LPLaunchListener): void`	进入回放房间
`bindPlayer(videoPlayer: IBJYVideoPlayer): void`	绑定主播放器
`bindCloudVideoPlayer(videoPlayer: IBJYVideoPlayer): void`	绑定云端视频
`` `getPlayer(): IBJYVideoPlayer \ ``	null`
`quitRoom(): void`	退出房间，释
`isPlaybackOffline(): boolean`	是否离线回放
`getRecordType(): number`	录制类型（0 大班 webrtc / webrtc / 3 合
`getTemplateType(): string`	模板类型字符

	“6”、”12”）
`isVideoMain(): boolean`	是否视频主区
`getRoomId(): number` / `getRoomToken(): string` / `getPlaybackId(): number`	房间元信息
`setOnSwitchPlaybackListener(listener: OnSwitchPlaybackListener): void`	合并回放切换

4.1.4. 进入房间

```
import { LPLaunchListener, PBRoom, PBLPError } from '@baijia/videoplayer';

class MyLaunchListener implements LPLaunchListener {
  onLaunchSteps(step: number, totalStep: number): void {
    // step / totalStep 即进房间进度百分比
  }
  onLaunchError(error: PBLPError): void { }
  onLaunchSuccess(room: PBRoom): void { }
}
pbRoom.enterRoom(new MyLaunchListener());
```

4.1.5. 绑定课件

鸿蒙端 PPT 组件 `PPTView`（`videoplayer/src/main/ets/playback/PlaybackPPT.ets`，从 `index.ets:61` 公开导出）由 `PPTController` 驱动：

```
import { PPTView, PPTController } from '@baijia/videoplayer';
```

`PPTController` 内部从 `PBRoom` 信令引擎接收翻页、画笔事件并驱动 `PPTView` 渲染。详细参数请参考 UI 模块中 `PBRoomPage` 的使用示例。

4.2. 聊天

```
pbRoom.getChatVM().getObservableOfNotifyDataCh
  .subscribe((messages: LPMessageModel[]) => {
    // 当前时刻的聊天消息集合
  });
```

ViewModel	接口	
`LPChatViewModel`	`getChatVM()`	
`LPDocListViewModel`	`getDocListVM()`	
`LPOnlineUsersViewModel`	`getOnlineUserVM()`	
`LPToolBoxViewModel`	`getToolBoxVM()`	

4.3. 获取在线人员

> *默认未开启在线人员信令，需通过 UI 层 `BJYPlayerSDK.enablePlaybackUserSignal = true` 启用（详见 UI 文档 §2.4）。*

```
pbRoom.getOnlineUserVM().getObservableOfOnline
  .subscribe((users: LPMessageUserModel[]) => {
    // 当前在线人员集合
  });
```

4.4. 监听视频开关状态

```
const recordType: number =
pbRoom.getRecordType();   // 0/1/2/3

pbRoom.getObservableOfVideoStatus()
  .subscribe((isVideoOn: boolean) => {
    if (recordType !== 1 && recordType !== 3) {
      // isVideoOn = false 时可显示占位图
    }
  });
```

4.5. 教室工具

4.5.1. 公告

```
pbRoom.getObservableOfAnnouncementChange()
  .subscribe((announcement:
IAnnouncementModel) => {
    announcement.getContent();
    announcement.getLink();
  });
```

4.5.2. 答题器

```
pbRoom.getToolBoxVM().getObservableOfAnswerSt
  .subscribe((answer: LPAnswerModel) => {
    answer.messageType;
    answer.type;
  });
```

4.5.3. 问答

```
pbRoom.getToolBoxVM().getObservableOfQuestionC
  .subscribe((items: LPQuestionPullListItem[]) =>
{ });
```

4.5.4. 红包（鸿蒙新增）

```
pbRoom.getObservableOfRedPackageConfig(playba
 roomId, token)
  .subscribe((config: RedPacketConfigBean) => {
});
pbRoom.receiveRedPackage(playbackId,
userNumber, userName, timeOffset, roomId,
token)
  .subscribe((red: RedPacketBean) => { });
```

4.6. 退出房间

```
// PPT 资源回收（如使用 PPTView）
pptController.destroy();
// 退出房间
pbRoom.quitRoom();
```

5. 下载

5.1. DownloadManager

`DownloadManager`（`videoplayer/src/main/ets/download/DownloadManager.ets`）使用单例模式：

```
import { DownloadManager } from
'@baijia/videoplayer';

```

```
const downloadManager =
DownloadManager.getInstance();      // line 77
```

5.1.1. 设置缓存路径（必需，必须在 `loadDownloadInfo` 之前）

```
downloadManager.setTargetFolder(context.filesDir
 + '/bb_video_downloaded/');   // line 90
```

5.1.2. 加载下载记录

```
// context 来自 UIAbilityContext / Context；
userIdentify、reload 均可选
await
downloadManager.loadDownloadInfo(context,
userIdentify?, reload?);       // line 119
```

> *鸿蒙端无 Android `Service` 概念，下载基于 `request` Kit + `taskpool`，应用进入后台后系统会限制网络任务时长，请在前台保活。*

5.1.3. 设置清晰度匹配规则

```
downloadManager.setPreferredDefinitionList([
  VideoDefinition.Audio,
  VideoDefinition._720P,
  VideoDefinition.SHD,
  VideoDefinition.HD,
  VideoDefinition.SD,
  VideoDefinition._1080P,
]);
// line 106
```

5.1.4. 获取下载记录 / 删除

```
const tasks: DownloadTask[] = downloadManager.getAllTasks();
downloadManager.deleteTask(task);
```

5.2. 点播下载

```
const task: DownloadTask = await downloadManager.newVideoDownloadTask(
  'video_filename',    // 文件名（非法字符自动替换）
  videoId,             // 点播 vid
  token,               // 视频 token
  'extraInfo',         // 透传字符串
  accessKey = '',      // 可选第三方鉴权 key
  isEncrypt = true     // 是否加密（仅当前下载有效）
);
// DownloadManager.ets line 300
```

5.3. 回放下载

```
const task: DownloadTask = await downloadManager.newPlaybackDownloadTask(
  'playback_filename',
  roomId,              // 房间 ID
  sessionId,           // 长期房间分段 ID，非长期传 0
  token,
  'extraInfo',
  version = -1,        // 裁剪版本，主版本传 -1
  isEncrypt = true
);
// DownloadManager.ets line 440
```

5.4. 下载状态回调

通过 `DownloadTask.setDownloadListener(listener)` 注册：

```
import { DownloadListener, DownloadTask } from '@baijia/videoplayer';

class MyDownloadListener implements DownloadListener {
  onProgress(task: DownloadTask): void { }
  onError(task: DownloadTask, e: Error): void { }
  onStarted(task: DownloadTask): void { }
  onPaused(task: DownloadTask): void { }
  onFinish(task: DownloadTask): void { }
  onDeleted(task: DownloadTask): void { }
}
task.setDownloadListener(new MyDownloadListener());
```

5.5. DownloadTask 接口

`download/DownloadModels.ets` ：

```
export interface DownloadTask {
  start(): void;
  pause(): void;
  restart(): void;
  cancel(): void;
  deleteFiles(): void;
  setDownloadListener(listener: DownloadListener): void;

  getVideoDownloadInfo(): DownloadModel;
```

```
10.  getSignalDownloadInfo(): DownloadModel | null;
11.  getTaskStatus(): TaskStatus;
12.  getSpeed(): number;           // byte/s
13.  getProgress(): number;
14.  getTotalLength(): number;      // 总字节数
15.  getDownloadedLength(): number;  // 已下载字节
     数
16.  getDownloadType(): DownloadType;
17.  getVideoFileName(): string;
18.  getSignalFileName(): string;
19.  getVideoDuration(): number;
20.  getVideoFilePath(): string;
21.  getSignalFilePath(): string;
22. }
```

枚举（ `DownloadModels.ets` ）：

枚举	值	说明	行号
`TaskStatus.New`	0	新建	98
`TaskStatus.Downloading`	1	下载中	99
`TaskStatus.Pause`	2	暂停	100
`TaskStatus.Error`	3	出错	101
`TaskStatus.Finish`	4	完成	102
`TaskStatus.Cancel`	5	已取消	103
`DownloadType.VIDEO`	0	点播	110
`DownloadType.PLAYBACK`	1	回放（视频 + 信令）	111
`FileType.VIDEO`	0	视频	118

`FileType.SIGNAL`	1	信令	119
`FileType.UNKNOWN`	2	未知	120
`FileType.AUDIO`	3	纯音频	121

5.6. DownloadModel 字段说明

`download/DownloadModels.ets:128` 起。主要字段：

```
class DownloadModel {
  videoId: number;          // vid
  sessionId: number;        // sessionId
  roomId: number;           // roomId
  url: string;              // 当前 CDN url
  definition: VideoDefinition;
  fileType: FileType;
  targetName: string;       // 保存的文件名
  targetFolder: string;     // 保存目录
  status: TaskStatus;
  totalLength: number;      // 总字节数
  downloadLength: number;   // 已下载字节数
  speed: number;            // 瞬时速度
  isEncrypt: boolean;
  videoToken: string;       // 视频 token
  extraInfo: string;        // 透传
  subtitleItems: SubtitleItem[];
  // 回放节点链表，链尾为信令信息；点播仅链头
  nextModel: DownloadModel | null;
}
```

6. 接口说明

6.1. PlayerStatus

player/PlayerStatus.ets:4

枚举	值	说明
STATE_IDLE	0	已创建未初始化
STATE_INITIALIZED	1	数据源已设置
STATE_PREPARED	2	已准备好，待播放
STATE_STARTED	3	播放中
STATE_PAUSED	4	暂停
STATE_STOPPED	5	停止
STATE_COMPLETED	6	播放结束
STATE_ERROR	7	出错

> ⚠ *命名与 Android 略有差异：Android 为 `STATE_PLAYBACK_COMPLETED`，鸿蒙为 `STATE_COMPLETED`。*

6.2. VideoDefinition

bean/VideoDefinition.ets:1

枚举	值	服务端键
UNKNOWN	-1	—
SD	0	low
HD	1	high
SHD	2	superHD
_720P	3	720p
_1080P	4	1080p

Audio	5	audio

辅助函数：

```
1. definitionFromString(type: string): VideoDefinition;  // line 29
2. definitionToString(def: VideoDefinition): string;  // line 37
```

6.3. BJYVideoInfo

bean/BJYVideoInfo.ets:1

```
1. export interface BJYVideoInfo {
2.   getVideoId(): number;
3.   getDuration(): number;                    // 服务端返回的时长（秒）
4.   getDefinition(): VideoDefinition;         // 默认清晰度
5.   getSupportedDefinitionList(): VideoDefinition[];  // 后端转码清晰度集合
6.   getVideoTitle(): string;
7.   getSubtitleItemList(): SubtitleItemInfo[];  // 字幕
8. }
```

SubtitleItemInfo 字段：id / videoId / name / url / zhUrl / enUrl / isDefault，提供静态工厂 fromJson(json) 。

6.4. VideoItem 关键字段

bean/VideoItem.ets，常用字段：videoId / duration / videoName / definition / playInfo /

`audioUrl` / `subtitleItems` / `waterMark` / `horseLamp` / `initPicture` / `startVideo` / `endVideo` / `roomId` / `sessionId` 。

CDN 信息 `CDNInfo` ： `cdn` / `definition` / `url` / `encUrl` / `width` / `height` / `size` / `weight` 。

7. 错误码

错误信息封装在 `LPError` （ `@baijia/livebase` ），核心字段： `code: number` / `message: string` 。

常见错误码沿用 Android 约定：

错误码	说明
-10000、-101	ijkplayer 内部错误（文件异常等）
10087	视频文件路径错误
10088	网络错误
-1	无网络连接
-2	移动网络播放
-4	未知错误
-5	没有找到对应清晰度
-6	离线播放传入非法路径
-7	离线播放对应路径的文件不存在
5101 / 5102 / 5103	token 异常
403	视频地址过期

8. 平台适配差异（鸿蒙 vs Android）

概念	Android
SDK 初始化	`BJYPlayerSDK.Builder(this).build()`
播放器创建	`VideoPlayerFactory.Builder()...build()`
视频视图	`BJYPlayerView extends FrameLayout`
视频比例	`AspectRatio` 枚举
异步流	`RxJava Observable`
PBRoom 工厂	`BJYPlayerSDK.newPlayBackRoom(ctx, ...)`
监听器实现	匿名类 / lambda
关键帧/字幕	`Observable<KeyFrameModel>`

9. 已知限制

1. **ArkTS 严格模式**：不允许 `any` / `unknown` / 匿名对象字面量实现接口，监听器必须用具名 class，`throw` 只能抛 `Error` 子类。
2. **`IjkMediaPlayer` 参数全部 string 类型**：例如 `seekTo(ms.toString())`、`setSpeed(\${rate}f`)`、`setVolume(vol.toString(), vol.toString())`。
3. **`@baijia/ijkplayer` 由 `@baijia/videoplayer` 传递依赖**，业务方无需单独声明，但 `XComponent` 必须正确

设置 `libraryname: 'ijkplayer_napi'`，否则找不到 native 库。

4. **`relationalStore` 无同步 API**，缓存/下载记录全部 `async`。
5. **JSON 字段名为 snake_case**（`play_info`、`cdn_list`、`enc_url`、`vod_default_definition` 等），在 `fromJson()` 中手动映射。
6. **`onPrepared` 后需手动 `start()`**：即使设置了 `start-on-prepared=1`，状态机需显式更新。
7. **后台播放限制**：鸿蒙系统对后台网络/媒体任务时长有限制，超长后台播放需要使用 ContinuousTask（持续任务）申请。