

1. 下载模块集成

视频下载功能通过单独模块 `media-player-sdk-addon-cache-down` 提供，需要添加对应依赖

```
{
  "dependencies": {
    "@polyvharmony/media-player-sdk-addon-cache-down": "$version"
  }
}
```

2. 视频下载

视频下载的管理类为 `PLVMediaDownloaderManager`，可以通过该类执行视频的下载、暂停、删除等操作

2.1 初始化

在使用视频下载功能前，需要调用初始化方法 `PLVMediaDownloaderManager.init`

```
/**
 * 初始化，调用其他方法前必须调用初始化方法
 */
async init(setting: PLVMediaDownloadSetting): Promise<void>
```

其中，可以通过 `setting` 参数配置下载的相关设置：

```

export class PLVMediaDownloadSetting {

    /**
     * 下载视频存放的根目录
     *
     * SDK 不会主动申请写入权限，外部调用需要确保 APP 具有对应目录的写入权限
     */
    readonly downloadRootDirectory: string;

    /**
     * 同时下载的任务数量限制
     */
    readonly concurrentDownloads: number;

    /**
     * 清晰度降级
     *
     * 无法下载指定的清晰度时，自动下载较低的清晰度
     */
    readonly allowBitRateFallback: boolean;

}

```

2.2 开始下载

首先，通过 `PLVMediaDownloaderManager.getDownloader` 获取对应视频的下载器

```

/**
 * 获取下载器
 *
 * @param mediaResource 视频资源
 * @param bitRate 指定下载的清晰度，可以通过 {@link updateSetting} 配置没有对应清晰度时自动降级。
 * 启用自动降级时可以通过 {@link IPLVMediaDownloaderListenerRegistry.downloadBitRate} 监听实际下载的清晰度。
 */
getDownloader(
    mediaResource: PLVMediaResource,
    bitRate: PLVMediaBitRate = PLVMediaBitRate.BITRATE_AUTO
): PLVMediaDownloader

```

然后，调用 `PLVMediaDownloaderManager.startDownloader` 传入对应的下载器，即开始视频的下载

```

/**
 * 开始下载
 */
startDownloader(downloader: PLVMediaDownloader): void

```

2.3 状态监听

下载过程中的进度、视频信息、下载速度等状态可以通过下载器的回调中心

`PLVMediaDownloader.listenerRegistry` 监听

```
const downloader = PLVMediaDownloaderManager.getInstance().getDownloader(...)
// 监听下载速度
downloader.listenerRegistry.downloadBytesPerSecond.observe((bytesPerSecond) => {
  // 业务操作
})
```

2.4 暂停、删除

视频下载的暂停和删除也是通过下载管理类 `PLVMediaDownloaderManager` 实现：

```
/**
 * 暂停下载
 */
pauseDownloader(downloader: PLVMediaDownloader): void

/**
 * 删除已下载的视频文件
 */
deleteDownloadContent(downloader: PLVMediaDownloader): void
```

3.播放离线视频

为了播放下载到本地的视频，需要在构造视频资源 `PLVMediaResource` 时，传入视频下载的根目录路径（即下载时配置的 `PLVMediaDownloadSetting.downloadRootDirectory`）

```
const mediaResource: PLVvodMediaResource = {
  // ...
  // 视频下载的路径
  localVideoSearchPaths: [downloadRootDirectory]
}
```

播放器会在 `localVideoSearchPaths` 下搜索对应的离线视频，在已下载对应视频到本地的情况下优先播放本地视频。

4.自定义传入token

视频下载支持加密视频版权保护的自定义传入token方式。生产环境建议由您的服务端生成下载所需的 token 后下发给客户端，再按如下方式传入：

```
downloadListenerRegistry.vodTokenRequestListener = {  
  async onRequestToken(mediaResource: PLVVodMediaResource): Promise<PLVVodVideoTokenVO> {  
    // 通过网络请求，向您的服务器请求视频下载token  
    const token: PLVVodVideoTokenVO = await fetchTokenFromNetwork()  
    // 将token返回给播放器  
    return token;  
  }  
}
```